C Aptitude

Note: All the programs are tested under Turbo C/C++ compilers.

It is assumed that,

- Programs run under DOS environment,
- The underlying machine is an x86 system,
- Program is compiled using Turbo C/C++ compiler.
- Proper and required header files are included,

The program output may depend on the information based on this assumptions (for example sizeof(int) == 2 may be assumed).

Predict the output or error(s) for the following:

1.
```
void main(){
        int  const * p=5;
        printf("%d",++(*p));
}
```

Answer:

Compiler error: Cannot modify a constant value.

Explanation:

p is a pointer to a "constant integer". But we tried to change the value of the "constant integer".

2. *main(){*

char s[ ]="man";

int i;

for(i=0;s[i];i++)

printf("\n%c%c%c%c",s[i],(s+i),*(i+s),i[s]);*

}

Answer:

mmmm

aaaa

nnnn

Explanation:

s[i], *(i+s), *(s+i), i[s] are all different ways of expressing the same idea. Generally array name is the base address for that array. Here **s** is the base address. **i** is the index number/displacement from the base address. So, indirecting it with * is same as s[i]. i[s] may be surprising. But in the case of C it is same as s[i].

3. *main(){*

```
    float me = 1.1;
    double you = 1.1;
    if(me==you)
        printf("I love U");
    else
        printf("I hate U");
}
```

Answer:

I hate U

Explanation:

For floating point numbers (float, double, long double) the values cannot be predicted exactly. Depending on the number of bytes, the precision with of the value represented varies. Float takes 4 bytes and long double takes 10 bytes. So float stores 0.9 with less precision than long double.

Rule of Thumb:

Never compare or at-least be cautious when using floating point numbers with relational operators (== , >, <, <=, >=,!=) .

4.
```
main() {
    static int var = 5;
    printf("%d ",var--);
    if(var)
        main();
```

}

Answer:

5 4 3 2 1

Explanation:

When *static* storage class is given, it is initialized once. The change in the value of a *static* variable is retained even between the function calls. Main is also treated like any other ordinary function, which can be called recursively.

5. *main(){*

int c[ ]={2.8,3.4,4,6.7,5};

*int j,*p=c,*q=c;*

for(j=0;j<5;j++) {

*printf(" %d ",*c);*

++q; }

for(j=0;j<5;j++){

*printf(" %d ",*p);*

++p; }

}

Answer:

2 2 2 2 2 2 3 4 6 5

Explanation:

Initially pointer c is assigned to both **p** and **q**. In the first loop, since only **q** is incremented and not c , the value 2 will be printed 5 times. In second loop **p** itself is incremented. So the values 2 3 4 6 5 will be printed.

6. *main(){*

extern int i;

i=20;

printf("%d",i);

}

Answer:

Linker Error : Undefined symbol '_i'

Explanation:

extern storage class in the following declaration,

extern int i;

specifies to the compiler that the memory for **i** is allocated in some other program and that address will be given to the current program at the time of linking. But linker finds that no other variable of name **i** is available in any other program with memory space allocated for it. Hence a linker error has occurred .

7. *main(){*

int i=-1,j=-1,k=0,l=2,m;

m=i++&&j++&&k++||l++;

printf("%d %d %d %d %d",i,j,k,l,m);

}

Answer:

0 0 1 3 1

Explanation :

Logical operations always give a result of **1 or 0** . And also the logical AND (&&) operator has higher priority over the logical OR (||) operator. So the expression '**i++ && j++ && k++'** is executed first. The result of this expression is 0 (-1 && -1 && 0 = 0). Now the expression is 0 || 2 which evaluates to 1 (because OR operator always gives 1 except for '0 || 0' combination- for which it gives 0). So the value of m is 1. The values of other variables are also incremented by 1.

8. *main(){*

*char *p;*

*printf("%d %d ",sizeof(*p),sizeof(p));*

}

Answer:

1 2

Explanation:

The *sizeof()* operator gives the number of bytes taken by its operand. P is a character pointer, which needs one byte for storing its value (a character). Hence sizeof(*p) gives a value of 1. Since it needs two bytes to store the address of the character pointer sizeof(p) gives 2.

9.
```
main(){
    int i=3;
    switch(i) {
        default:printf("zero");
        case 1: printf("one");
                break;
        case 2:printf("two");
                break;
        case 3: printf("three");
                break;
     }
}
```

Answer :

three

Explanation :

The *default* case can be placed anywhere inside the loop. It is executed only when all other cases doesn't match.

10.
```
main(){
    printf("%x",-1<<4);
}
```

Answer:

fff0

Explanation :

-1 is internally represented as all 1's. When left shifted four times the least significant 4 bits are filled with 0's.The %x format specifier specifies that the integer value be printed as a hexadecimal value.

11.
```
main(){
    char string[]="Hello World";
    display(string);
}
void display(char *string){
    printf("%s",string);
}
```

Answer:

Compiler Error : Type mismatch in redeclaration of function display

Explanation :

In third line, when the function **display** is encountered, the compiler doesn't know anything about the function **display**. It assumes the arguments and return types to be integers, (which is the default type). When it sees the actual function **display,** the arguments and type contradicts with what it has assumed previously. Hence a compile time error occurs.

12.
```
main(){
    int c=- -2;
```

printf("c=%d",c);

}

Answer:

c=2;

Explanation:

Here *unary minus* (or negation) operator is used twice. Same maths rules applies, ie. minus * minus= plus.

Note:

However you cannot give like --2. Because -- operator can only be applied to variables as a **decrement** operator (eg., i--). 2 is a constant and not a variable.

13. *#define int char*

main(){

int i=65;

printf("sizeof(i)=%d",sizeof(i));

}

Answer:

sizeof(i)=1

Explanation:

Since the *#define* replaces the string **int** by the macro **char**

14. *main(){*

int i=10;

```
    i=!i>14;
    printf("i=%d",i);
}
```

Answer:

i=0

Explanation:

In the expression **!i>14**, NOT (!) operator has more precedence than ‘ >’ symbol. **!** is a unary logical operator. !i (!10) is 0 (not of true is false). 0>14 is false (zero).

15.
```
main(){
    char s[]={'a','b','c','\n','c','\0'};
    char *p,*str,*str1;
    p=&s[3];
    str=p;
    str1=s;
    printf("%d",++*p + ++*str1-32);
}
```

Answer:

77

Explanation:

p is pointing to character '\n'. str1 is pointing to character 'a' ++*p. "p is pointing to '\n' and that is incremented by one." the ASCII value of '\n' is 10, which is then

incremented to 11. The value of ++*p is 11. ++*str1, str1 is pointing to 'a' that is incremented by 1 and it becomes 'b'. ASCII value of 'b' is 98.

Now performing (11 + 98 – 32), we get 77 (77 is the ASCII value for "M");

So we get the output 77.

16. *main(){*

```
int a[2][2][2] = { {10,2,3,4}, {5,6,7,8}  };
int *p,*q;
p=&a[2][2][2];
*q=***a;
printf("%d----%d",*p,*q);
```

}

Answer:

SomeGarbageValue---1

Explanation:

p=&a[2][2][2] you declare only two 2D arrays, but you are trying to access the third 2D(which you are not declared) it will print garbage values. *q=***a starting address of a is assigned integer pointer. Now q is pointing to starting address of a. If you print *q, it will print first element of 3D array.

17. *main(){*

```
struct xx{
    char name[]="hello";
```

```
};
struct xx *s;
printf("%s",s->name);
}
```

Answer:

Compiler Error

Explanation:

You should not initialize variables in structure declaration.

18.
```
main(){
struct xx{
int x;
struct yy{
char s;
struct xx *p;
};
struct yy *q;
};
}
```

Answer:

No output.

Explanation:

Pointer to the same type of structures are known as *self referential structures*. They are particularly used in implementing datastructures like trees. Structures within structures are known as n*ested structures.*

19. *main()*

{

printf("\nab");

printf("\bsi");

printf("\rha");

}

Answer:

hai

Explanation:

\n - newline

\b - backspace

\r - linefeed

20. *main()*

{

int i=5;

printf("%d%d%d%d%d%d",i++,i--,++i,--i,i);

}

Answer:

45545

Explanation:

The arguments in a function call are pushed into the stack from left to right. The evaluation is by popping out from the stack. and the evaluation is from right to left, hence the result.

21. *#define square(x) x*x*

main()

{

int i;

i = 64/square(4);

printf("%d",i);

}

Answer:

64

Explanation:

The macro call square(4) will substituted by 4*4 so the expression becomes i = 64/4*4 . Since / and * has equal priority the expression will be evaluated as (64/4)*4 i.e. 16*4 = 64

22. *main()*

{

*char *p="hai friends",*p1;*

```
    p1=p;
    while(*p!='\0') ++*p++;
    printf("%s   %s",p,p1);
}
```

Answer:

ibj!gsjfoet

Explanation:

++*p++ will be parse in the given order

- *p that is value at the location currently pointed by p will be taken
- ++*p the retrieved value will be incremented
- when ; is encountered the location will be incremented, that is p++ will be executed

Hence, in the while loop initial value pointed by p is ‘h’, which is changed to ‘i’ by executing ++*p and pointer moves to point, ‘a’ which is similarly changed to ‘b’ and so on. Similarly blank space is converted to ‘!’. Thus, we obtain value in p becomes “ibj!gsjfoet” and since p reaches ‘\0’ and p1 points to p thus p1doesnot print anything.

23.
```
#define a 10
main()
{
    #define a 50
    printf("%d",;
}
```

Answer:

50

Explanation:

The *preprocessor* directives can be redefined anywhere in the program. So the most recently assigned value will be taken.

24. *#define clrscr() 100*

main(){

clrscr();

printf("%d\n",clrscr());

}

Answer:

100

Explanation:

Preprocessor executes as a seperate pass before the execution of the compiler. So textual replacement of clrscr() to 100 occurs.The input program to compiler looks like this :

```
main()
{
    100;
    printf("%d\n",100);
}
```

Note:

100; is an executable statement but with no action. So it doesn't give any problem.

25. *main(){*

printf("%p",main);

}

Answer:

Some address will be printed.

Explanation:

Function names are just addresses (just like array names are addresses).main() is also a function. So the address of function main will be printed. %p in printf specifies that the argument is an address. They are printed as hexadecimal numbers.

26. *main(){*

clrscr();

}

clrscr();

Answer:

No output/error

Explanation:

The first clrscr() occurs inside a function. So it becomes a function call. In the second clrscr(); is a function declaration (because it is not inside any function).

27. *enum colors {BLACK,BLUE,GREEN}*

main(){

printf("%d..%d..%d",BLACK,BLUE,GREEN);

return(1);

}

Answer:

0..1..2

Explanation:

enum assigns numbers starting from 0, if not explicitly defined.

28. *void main(){*

*char far *farther,*farthest;*

printf("%d..%d",sizeof(farther),sizeof(farthest));

}

Answer:

4..2

Explanation:

The second pointer is of char type and not a far pointer

29. *main(){*

int i=400,j=300;

printf("%d..%d");

}

Answer:

400..300

Explanation:

printf takes the values of the first two assignments of the program. Any number of *printf*'s may be given. All of them take only the first two values. If more number of assignments given in the program,then *printf* will take garbage values.

30.
```
main(){
    char *p;
    p="Hello";
    printf("%c\n",*&*p);
}
```

Answer:

H

Explanation:

* is a dereference operator & is a reference operator. They can be applied any number of times provided it is meaningful. Here p points to the first character in the string "Hello". *p dereferences it and so its value is H. Again & references it to an address and * dereferences it to the value H.

31.
```
main(){
    int i=1;
    while (i<=5){
```

```
        printf("%d",i);
        if (i>2)
                goto here;
        i++;
        }
}
fun(){
        here:
        printf("PP");
}
```

Answer:

Compiler error: Undefined label 'here' in function main

Explanation:

Labels have functions *scope*, in other words The *scope* of the labels is limited to functions . The *label* 'here' is available in function fun() Hence it is not visible in function main.

32.
```
main(){
        static char names[5][20]={"pascal","ada","cobol","fortran","perl"};
        int i;
        char *t;
        t=names[3];
        names[3]=names[4];
```

```
        names[4]=t;
        for (i=0;i<=4;i++)
            printf("%s",names[i]);
}
```

Answer:

Compiler error: Lvalue required in function main

Explanation:

Array names are *pointer constants*. So it cannot be modified.

33.
```
void main(){
        int i=5;
        printf("%d",i++ + ++i);
}
```

Answer:

Output Cannot be predicted exactly.

Explanation:

Side effects are involved in the evaluation of i.

34.
```
void main(){
        int i=5;
        printf("%d",i+++++i);
}
```

Answer:

Compiler Error

Explanation:

The expression i+++++i is parsed as i ++ ++ + i which is an illegal combination of operators.

35. *main(){*

int i=1,j=2;

switch(i){

case 1: printf("GOOD");

break;

case j: printf("BAD");

break;

}

}

Answer:

Compiler Error: Constant expression required in function main.

Explanation:

The case statement can have only constant expressions (this implies that we cannot use variable names directly so an error).

Note:

Enumerated types can be used in case statements.

36. *main(){*

```
    int i;
    printf("%d",scanf("%d",&i));  // value 10 is given as input here
}
```

Answer:

1

Explanation:

Scanf returns number of items successfully read.Here 10 is given as input which should have been scanned successfully. So number of items read is 1.

37.
```
#define f(g,g2) g##g2
main(){
    int var12=100;
    printf("%d",f(var,12));
}
```

Answer:

100

38.
```
main(){
    int i=0;
    for(;i++;printf("%d",i)) ;
        printf("%d",i);
}
```

Answer:

1

Explanation:

Before entering into the for loop the checking condition is "evaluated". Here it evaluates to 0 (false) and comes out of the loop, and i is incremented (note the semicolon after the for loop).

39. *main(){*

extern int i;

i=20;

printf("%d",sizeof(i));

}

Answer:

Linker error: undefined symbol '_i'.

Explanation:

extern declaration specifies that the variable i is defined somewhere else. The compiler passes the external variable to be resolved by the linker. So compiler doesn't find an error. During linking the linker searches for the definition of i. Since it is not found the linker flags an error.

40. *main(){*

printf("%d", out);

}

int out=100;

Answer:

Compiler error: undefined symbol out in function main.

Explanation:

The rule is that a variable is available for use from the point of declaration. Even though a is a global variable, it is not available for main. Hence an error.

41. *main(){*

extern out;

printf("%d", out);

}

int out=100;

Answer:

100

Explanation:

This is the correct way of writing the previous program.

42. *main(){*

show();

}

void show(){

printf("I'm the greatest");

}

Answer:

Compier error: Type mismatch in redeclaration of show.

Explanation:

When the compiler sees the function show it doesn't know anything about it. So the default return type (ie, int) is assumed. But when compiler sees the actual definition of show mismatch occurs since it is declared as void. Hence the error.

The solutions are as follows:

1. declare void show() in main() .
2. define show() before main().
3. declare extern void show() before the use of show().

43. *main( ){*

int a[2][3][2] = {{{2,4},{7,8},{3,4}},{{2,2},{2,3},{3,4}}};

*printf("%u %u %u %d \n",a,*a,**a,***;*

*printf("%u %u %u %d \n",a+1,*a+1,**a+1,***a+1);*

}

Answer:

100, 100, 100, 2

114, 104, 102, 3

Explanation:

The given array is a 3-D one. It can also be viewed as a 1-D array.

2	4	7	8	3	4	2	2	2	3	3	4
100	102	104	106	108	110	112	114	116	118	120	122

Thus, for the first printf statement a, *a, **a give address of first element. Since the indirection ***a gives the value. Hence, the first line of the output. For the second printf a+1 increases in the third dimension thus points to value at 114, *a+1 increments in second dimension thus points to 104, **a +1 increments the first dimension thus points to 102 and ***a+1 first gets the value at first location and then increments it by 1. Hence, the output.

44.
```
main( ){
    int a[ ] = {10,20,30,40,50},j,*p;
    for(j=0; j<5; j++){
        printf("%d" ,*;
        a++;
    }
    p = a;
    for(j=0; j<5; j++){
        printf("%d " ,*p);
        p++;
    }
}
```

Answer:

Compiler error: lvalue required.

Explanation:

Error is in line with statement a++. The operand must be an lvalue and may be of any of scalar type for the any operator, array name only when subscripted is an lvalue. Simply array name is a non-modifiable lvalue.

45. *main( ){*

```
static int  a[ ]   = {0,1,2,3,4};
int  *p[ ] = {a,a+1,a+2,a+3,a+4};
int  **ptr =  p;
ptr++;
printf("\n %d  %d  %d", ptr-p, *ptr-a, **ptr);
*ptr++;
printf("\n %d  %d  %d", ptr-p, *ptr-a, **ptr);
*++ptr;
printf("\n %d  %d  %d", ptr-p, *ptr-a, **ptr);
++*ptr;
printf("\n %d  %d  %d", ptr-p, *ptr-a, **ptr);
```

}

Answer:

111

222

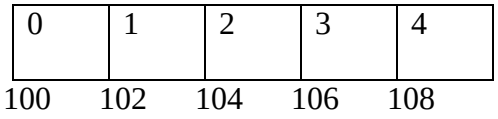
333

344

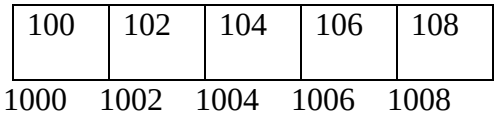
Explanation:

Let us consider the array and the two pointers with some address

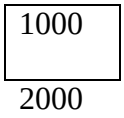
a



p



ptr



After execution of the instruction ptr++ value in ptr becomes 1002, if scaling factor for integer is 2 bytes. Now ptr – p is value in ptr – starting location of array p, (1002 – 1000) / (scaling factor) = 1, *ptr – a = value at address pointed by ptr – starting value of array a, 1002 has a value 102 so the value is (102 – 100)/(scaling factor) = 1, **ptr is the value stored in the location pointed by the pointer of ptr = value pointed by value pointed by 1002 = value pointed by 102 = 1. Hence the output of the firs printf is 1, 1, 1.

After execution of *ptr++ increments value of the value in ptr by scaling factor, so it becomes1004. Hence, the outputs for the second printf are ptr – p = 2, *ptr – a = 2, **ptr = 2.

After execution of *++ptr increments value of the value in ptr by scaling factor, so it becomes1004. Hence, the outputs for the third printf are ptr – p = 3, *ptr – a = 3, **ptr = 3.

After execution of ++*ptr value in ptr remains the same, the value pointed by the value is incremented by the scaling factor. So the value in array p at location 1006 changes from 106 10 108,. Hence, the outputs for the fourth printf are ptr – p = 1006 – 1000 = 3, *ptr – a = 108 – 100 = 4, **ptr = 4.

46. *main( ){*

*char *q;*

int j;

for (j=0; j<3; j++) scanf("%s" ,(q+j));

for (j=0; j<3; j++) printf("%c" ,(q+j));*

for (j=0; j<3; j++) printf("%s" ,(q+j));

}

Explanation:

Here we have only one pointer to type char and since we take input in the same pointer thus we keep writing over in the same location, each time shifting the pointer value by 1. Suppose the inputs are MOUSE, TRACK and VIRTUAL. Then for the first input suppose the pointer starts at location 100 then the input one is stored as

M	O	U	S	E	\0

When the second input is given the pointer is incremented as j value becomes 1, so the input is filled in memory starting from 101.

M	T	R	A	C	K	\0

The third input starts filling from the location 102

M	T	V	I	R	T	U	A	L	\0

This is the final value stored .

The first printf prints the values at the position q, q+1 and q+2 = M T V

The second printf prints three strings starting from locations q, q+1, q+2

i.e MTVIRTUAL, TVIRTUAL and VIRTUAL.

47. *main( ){*

*void *vp;*

*char ch = 'g', *cp = "goofy";*

int j = 20;

vp = &ch;

*printf("%c", *(char *)vp);*

vp = &j;

printf("%d",(int *)vp);*

vp = cp;

*printf("%s",(char *)vp + 3);*

}

Answer:

g20fy

Explanation:

Since a void pointer is used it can be type casted to any other type pointer. vp = &ch stores address of char ch and the next statement prints the value stored in vp after type casting it to the proper data type pointer. the output is 'g'. Similarly the output from

second printf is ‘20’. The third printf statement type casts it to print the string from the 4th value hence the output is ‘fy’.

48.
```
main ( ){
static char *s[ ]  = {“black”, “white”, “yellow”, “violet”};
char **ptr[ ] = {s+3, s+2, s+1, s}, ***p;
p = ptr;
**++p;
printf(“%s”,*--*++p + 3);
}
```

Answer:

ck

Explanation:

In this problem we have an array of char pointers pointing to start of 4 strings. Then we have ptr which is a pointer to a pointer of type char and a variable p which is a pointer to a pointer to a pointer of type char. p hold the initial value of ptr, i.e. p = s+3. The next statement increment value in p by 1 , thus now value of p = s+2. In the printf statement the expression is evaluated *++p causes gets value s+1 then the pre decrement is executed and we get s+1 – 1 = s . the indirection operator now gets the value from the array of s and adds 3 to the starting address. The string is printed starting from this position. Thus, the output is ‘ck’.

49.
```
main(){
```

```
    int  i, n;
    char *x = "girl";
    n = strlen(x);
    *x = x[n];
    for(i=0; i<n; ++i){
        printf("%s\n",x);
        x++;
    }
}
```

Answer:

(blank space)

irl

rl

l

Explanation:

Here a string (a pointer to char) is initialized with a value "girl". The strlen function returns the length of the string, thus n has a value 4. The next statement assigns value at the nth location ('\0') to the first location. Now the string becomes "\0irl" . Now the printf statement prints the string after each iteration it increments it starting position. Loop starts from 0 to 4. The first time x[0] = '\0' hence it prints nothing and pointer value is incremented. The second time it prints from x[1] i.e "irl" and the third time it prints "rl" and the last time it prints "l" and the loop terminates.

50. *int i,j;*

for(i=0;i<=10;i++){

j+=5;

assert(i<5);

}

Answer:

Runtime error: Abnormal program termination.

assert failed (i<5), <file name>,<line number>

Explanation:

asserts are used during debugging to make sure that certain conditions are satisfied. If assertion fails, the program will terminate reporting the same. After debugging use,

#undef NDEBUG

and this will disable all the assertions from the source code. Assertion is a good debugging tool to make use of.

51. *main(){*

int i=-1;

+i;

printf("i = %d, +i = %d \n",i,+i);

}

Answer:

i = -1, +i = -1

Explanation:

Unary + is the only dummy operator in C. Where-ever it comes you can just ignore it just because it has no effect in the expressions (hence the name dummy operator).

52. What are the files which are automatically opened when a C file is executed?

Answer:

stdin, stdout, stderr (standard input,standard output,standard error).

53. What will be the position of the file marker?

a) fseek(ptr,0,SEEK_SET);

b) fseek(ptr,0,SEEK_CUR);

Answer :

a) The SEEK_SET sets the file position marker to the starting of the file.

b) The SEEK_CUR sets the file position marker to the current position of the file.

54. main(){

char name[10],s[12];

scanf(" \"%[^\"]\"",s);

}

How scanf will execute?

Answer:

First it checks for the leading white space and discards it.Then it matches with a quotation mark and then it reads all character upto another quotation mark.

55. *What is the problem with the following code segment?*

while ((fgets(receiving array,50,file_ptr)) != EOF) ;

Answer & Explanation:

fgets returns a pointer. So the correct end of file check is checking for != NULL.

56. *main(){*

main();

}

Answer:

Runtime error : Stack overflow.

Explanation:

main function calls itself again and again. Each time the function is called its return address is stored in the call stack. Since there is no condition to terminate the function call, the call stack overflows at runtime. So it terminates the program and results in an error.

57. *main(){*

*char *cptr,c;*

*void *vptr,v;*

```
    c=10;  v=0;
    cptr=&c; vptr=&v;
    printf("%c%v",c,v);
}
```

Answer:

Compiler error (at line number 4): size of v is Unknown.

Explanation:

You can create a variable of type void * but not of type void, since void is an empty type. In the second line you are creating variable vptr of type void * and v of type void hence an error.

58.
```
main() {
    char *str1="abcd";
    char str2[]="abcd";
    printf("%d %d %d",sizeof(str1),sizeof(str2),sizeof("abcd"));
}
```

Answer:

2 5 5

Explanation:

In first sizeof, str1 is a character pointer so it gives you the size of the pointer variable. In second sizeof the name str2 indicates the name of the array whose size is 5 (including the '\0' termination character). The third sizeof is similar to the second one.

59.
```
main(){
    char not;
    not=!2;
    printf("%d",not);
}
```

Answer:

0

Explanation:

! is a logical operator. In C the value 0 is considered to be the boolean value FALSE, and any non-zero value is considered to be the boolean value TRUE. Here 2 is a non-zero value so TRUE. !TRUE is FALSE (0) so it prints 0.

60.
```
#define FALSE -1
#define TRUE   1
#define NULL   0
main(){
    if(NULL)
        puts("NULL");
    else if(FALSE)
        puts("TRUE");
    else
        puts("FALSE");
  }
```

Answer:

TRUE

Explanation:

The input program to the compiler after processing by the preprocessor is,

```
main(){
    if(0)
        puts("NULL");
    else if(-1)
        puts("TRUE");
    else
        puts("FALSE");
}
```

Preprocessor doesn't replace the values given inside the double quotes. The check by if condition is boolean value false so it goes to else. In second if -1 is boolean value true hence "TRUE" is printed.

61.
```
main(){
    int k=1;
    printf("%d==1 is ""%s",k,k==1?"TRUE":"FALSE");
}
```

Answer:

1==1 is TRUE

Explanation:

When two strings are placed together (or separated by white-space) they are concatenated (this is called as "stringization" operation). So the string is as if it is given as "%d==1 is %s". The conditional operator(?:) evaluates to "TRUE".

62.
```
main(){
    int y;
    scanf("%d",&y); // input given is 2000
    if( (y%4==0 && y%100 != 0) || y%100 == 0 )
        printf("%d is a leap year");
    else
        printf("%d is not a leap year");
}
```

Answer:

2000 is a leap year

Explanation:

An ordinary program to check if leap year or not.

63.
```
#define max 5
#define int arr1[max]
main(){
    typedef char arr2[max];
    arr1 list={0,1,2,3,4};
    arr2 name="name";
```

```
    printf("%d %s",list[0],name);
}
```

Answer:

Compiler error (in the line arr1 list = {0,1,2,3,4})

Explanation:

arr2 is declared of type array of size 5 of characters. So it can be used to declare the variable name of the type arr2. But it is not the case of arr1. Hence an error.

Rule of Thumb:

#defines are used for textual replacement whereas typedefs are used for declaring new types.

64.

```
int i=10;
main() {
    extern int i; {
        int i=20;{
            const volatile unsigned i=30;
            printf("%d",i);
        }
        printf("%d",i);
    }
    printf("%d",i);
}
```

Answer:

30,20,10

Explanation:

'{' introduces new block and thus new scope. In the innermost block i is declared as, const volatile unsigned which is a valid declaration. i is assumed of type int. So printf prints 30. In the next block, i has value 20 and so printf prints 20. In the outermost block, i is declared as extern, so no storage space is allocated for it. After compilation is over the linker resolves it to global variable i (since it is the only variable visible there). So it prints i's value as 10.

65.
```
main(){
    int *j;{
        int i=10;
        j=&i;
    }
    printf("%d",*j);
}
```

Answer:

10

Explanation:

The variable i is a block level variable and the visibility is inside that block only. But the lifetime of i is lifetime of the function so it lives upto the exit of main function. Since the i is still allocated space, *j prints the value stored in i since j points i.

66. *main(){*

int i=-1;

-i;

printf("i = %d, -i = %d \n",i,-i);

}

Answer:

i = -1, -i = 1

Explanation:

-i is executed and this execution doesn't affect the value of i. In printf first you just print the value of i. After that the value of the expression -i = -(-1) is printed.

67. *main() {*

const int i=4;

float j;

j = ++i;

printf("%d %f", i,++j);

}

Answer:

Compiler error

Explanation:

i is a constant. you cannot change the value of constant

68. *main(){*

int a[2][2][2] = { {10,2,3,4}, {5,6,7,8} };

*int *p,*q;*

p=&a[2][2][2];

q=*a;*

*printf("%d..%d",*p,*q);*

}

Answer:

garbagevalue..1

Explanation:

p=&a[2][2][2] you declare only two 2D arrays. but you are trying to access the third 2D(which you are not declared) it will print garbage values. *q=***a starting address of a is assigned integer pointer. now q is pointing to starting address of a.if you print *q meAnswer:it will print first element of 3D array.

69. *main() {*

register i=5;

char j[]= "hello";

printf("%s %d",j,i);

}

Answer:

hello 5

Explanation:

if you declare i as register compiler will treat it as ordinary integer and it will take integer value. i value may be stored either in register or in memory.

70.
```
main(){
    int i=5,j=6,z;
    printf("%d",i+++j);
 }
```

Answer:

11

Explanation:

The expression i+++j is treated as (i++ + j).

71.
```
struct aaa{
    struct aaa *prev;
    int i;
    struct aaa *next;
};
main(){
    struct aaa abc,def,ghi,jkl;
    int x=100;
    abc.i=0;abc.prev=&jkl;
    abc.next=&def;
    def.i=1;def.prev=&abc;def.next=&ghi;
```

ghi.i=2;ghi.prev=&def;

ghi.next=&jkl;

jkl.i=3;jkl.prev=&ghi;jkl.next=&abc;

x=abc.next->next->prev->next->i;

printf("%d",x);

}

Answer:

2

Explanation:

above all statements form a double circular linked list;

abc.next->next->prev->next->i

this one points to "ghi" node the value of at particular node is 2.

72. *struct point{*

int x;

int y;

};

*struct point origin,*pp;*

main(){

pp=&origin;

*printf("origin is(%d%d)\n",(*pp).x,(*pp).y);*

printf("origin is (%d%d)\n",pp->x,pp->y);

}

Answer:

origin is(0,0)

origin is(0,0)

Explanation:

pp is a pointer to structure. we can access the elements of the structure either with arrow mark or with indirection operator.

Note:

Since structure point is globally declared x & y are initialized as zeroes

73. *main(){*

int i=_l_abc(10);

printf("%d\n",--i);

}

int _l_abc(int i){

return(i++);

}

Answer:

9

Explanation:

return(i++) it will first return i and then increments. i.e. 10 will be returned.

74. *main()*{

*char *p;*

```
    int *q;
    long *r;
    p=q=r=0;
    p++;
    q++;
    r++;
    printf("%p...%p...%p",p,q,r);
}
```

Answer:

0001...0002...0004

Explanation:

++ operator when applied to pointers increments address according to their corresponding data-types.

75.
```
main(){
    char c=' ',x,convert(z);
    getc(c);
    if((c>='a') && (c<='z'))
        x=convert(c);
    printf("%c",x);
}
convert(z){
    return z-32;
```

}

Answer:

Compiler error

Explanation:

Declaration of convert and format of getc() are wrong.

76. *main(int argc, char **argv){*

printf("enter the character");

getchar();

sum(argv[1],argv[2]);

}

sum(num1,num2)int num1,num2;{

return num1+num2;

}

Answer:

Compiler error.

Explanation:

argv[1] & argv[2] are strings. They are passed to the function sum without converting it to integer values.

77. *int one_d[]={1,2,3};*

main(){

*int *ptr;*

```
        ptr=one_d;
        ptr+=3;
        printf("%d",*ptr);
}
```

Answer:

garbage value

Explanation:

ptr pointer is pointing to out of the array range of one_d.

78.
```
aaa() {
        printf("hi");
}
bbb(){
        printf("hello");
}
ccc(){
        printf("bye");
}
main(){
        int (*ptr[3])();
        ptr[0]=aaa;
        ptr[1]=bbb;
        ptr[2]=ccc;
```

ptr[2]();

}

Answer:

bye

Explanation:

ptr is array of pointers to functions of return type int.ptr[0] is assigned to address of the function aaa. Similarly ptr[1] and ptr[2] for bbb and ccc respectively. ptr[2]() is in effect of writing ccc(), since ptr[2] points to ccc.

79. *main(){*

*FILE *ptr;*

char i;

ptr=fopen("zzz.c","r");

while((i=fgetch(ptr))!=EOF)

printf("%c",i);

}

Answer:

contents of zzz.c followed by an infinite loop

Explanation:

The condition is checked against EOF, it should be checked against NULL.

80. *main(){*

int i =0;j=0;

```
    if(i && j++)
        printf("%d..%d",i++,j);
    printf("%d..%d,i,j);
}
```

Answer:

0..0

Explanation:

The value of i is 0. Since this information is enough to determine the truth value of the boolean expression. So the statement following the if statement is not executed. The values of i and j remain unchanged and get printed.

81.
```
main(){
    int i;
    i = abc();
    printf("%d",i);
}
abc(){
    _AX = 1000;
}
```

Answer:

1000

Explanation:

Normally the return value from the function is through the information from the accumulator. Here _AH is the pseudo global variable denoting the accumulator. Hence, the value of the accumulator is set 1000 so the function returns value 1000.

82. *int i;*

main(){

int t;

for (t=4;scanf("%d",&i)-t;printf("%d\n",i))

printf("%d--",t--);

}

// If the inputs are 0,1,2,3 find the o/p

Answer:

4--0

3--1

2--2

Explanation:

Let us assume some x= scanf("%d",&i)-t the values during execution will be,

t	i	x
4	0	-4
3	1	-2
2	2	0

83. *main(){*

```
int a= 0;int b = 20;char x =1;char y =10;
if(a,b,x,y)
printf("hello");
}
```

Answer:

hello

Explanation:

The comma operator has associativity from left to right. Only the rightmost value is returned and the other values are evaluated and ignored. Thus the value of last variable y is returned to check in if. Since it is a non zero value if becomes true so, "hello" will be printed.

84.
```
main(){
unsigned int i;
for(i=1;i>-2;i--)
    printf("c aptitude");
}
```

Explanation:

i is an unsigned integer. It is compared with a signed value. Since the both types doesn't match, signed is promoted to unsigned value. The unsigned equivalent of -2 is a huge value so condition becomes false and control comes out of the loop.

85. *In the following pgm add a stmt in the function fun such that the address of 'a' gets stored in 'j'.*

```
main(){
        int * j;
        void fun(int **);
        fun(&j);
}
void fun(int **k) {
        int a =0;
        /* add a stmt here*/
}
```

Answer:

*k = &a

Explanation:

The argument of the function is a pointer to a pointer.

86. *What are the following notations of defining functions known as?*

i.
```
int abc(int a,float b) {
    /* some code */
    }
```

ii.
```
int abc(a,b)
int a; float b; {
    /* some code*/
```

}

Answer:

i. ANSI C notation

ii. Kernighan & Ritche notation

87. *main(){*

*char *p;*

p="%d\n";

p++;

p++;

printf(p-2,300);

}

Answer:

300

Explanation:

The pointer points to % since it is incremented twice and again decremented by 2, it points to '%d\n' and 300 is printed.

88. *main(){*

char a[100];

a[0]='a';a[1]]='b';a[2]='c';a[4]='d';

abc(;

}

```
abc(char a[]){
        a++;
        printf("%c",*;
        a++;
        printf("%c",*;
}
```

Explanation:

The base address is modified only in function and as a result a points to 'b' then after incrementing to 'c' so bc will be printed.

89.
```
func(a,b)
int a,b;{
        return( a= (a==b) );
}
main(){
        int process(),func();
        printf("The value of process is %d !\n ",process(func,3,6));
}
process(pf,val1,val2)
int (*pf) ();
int val1,val2;{
        return((*pf) (val1,val2));
 }
```

Answer:

The value if process is 0 !

Explanation:

The function 'process' has 3 parameters - 1, a pointer to another function 2 and 3, integers. When this function is invoked from main, the following substitutions for formal parameters take place: func for pf, 3 for val1 and 6 for val2. This function returns the result of the operation performed by the function 'func'. The function func has two integer parameters. The formal parameters are substituted as 3 for a and 6 for b. since 3 is not equal to 6, a==b returns 0. therefore the function returns 0 which in turn is returned by the function 'process'.

90.
```
void main(){
    static int i=5;
    if(--i){
        main();
        printf("%d ",i);
    }
}
```

Answer:

0 0 0 0

Explanation:

The variable "I" is declared as static, hence memory for I will be allocated for only once, as it encounters the statement. The function main() will be called recursively

unless I becomes equal to 0, and since main() is recursively called, so the value of static I ie., 0 will be printed every time the control is returned.

91. *void main(){*

int k=ret(sizeof(float));

printf("\n here value is %d",++k);

}

int ret(int ret){

ret += 2.5;

return(ret);

}

Answer:

Here value is 7

Explanation:

The int ret(int ret), ie., the function name and the argument name can be the same.

Firstly, the function ret() is called in which the sizeof(float) ie., 4 is passed, after the first expression the value in ret will be 6, as ret is integer hence the value stored in ret will have implicit type conversion from float to int. The ret is returned in main() it is printed after and preincrement.

92. *void main(){*

char a[]="12345\0";

int i=strlen(;

```
    printf("here in 3 %d\n",++i);
}
```

Answer:

here in 3 6

Explanation:

The char array 'a' will hold the initialized string, whose length will be counted from 0 till the null character. Hence the 'I' will hold the value equal to 5, after the pre-increment in the printf statement, the 6 will be printed.

93.
```
void main(){
    unsigned giveit=-1;
    int gotit;
    printf("%u ",++giveit);
    printf("%u \n",gotit=--giveit);
}
```

Answer:

0 65535

94.
```
void main(){
    int i;
    char a[]="\0";
    if(printf("%s\n",)
        printf("Ok here \n");
```

```
        else
                printf("Forget it\n");
}
```

Answer:

Ok here

Explanation:

Printf will return how many characters does it print. Hence printing a null character returns 1 which makes the if statement true, thus "Ok here" is printed.

95.
```
void main(){
        void *v;
        int integer=2;
        int *i=&integer;
        v=i;
        printf("%d",(int*)*v);
}
```

Answer:

Compiler Error. We cannot apply indirection on type void*.

Explanation:

Void pointer is a generic pointer type. No pointer arithmetic can be done on it. Void pointers are normally used for,

1. Passing generic pointers to functions and returning such pointers.
2. As a intermediate pointer type.

3. Used when the exact pointer type will be known at a later point of time.

96. *void main(){*

int i=i++,j=j++,k=k++;

printf("%d%d%d",i,j,k);

}

Answer:

Garbage values.

Explanation:

An identifier is available to use in program code from the point of its declaration.

So expressions such as i = i++ are valid statements. The i, j and k are automatic variables and so they contain some garbage value. *Garbage in is garbage out (GIGO).*

97. *void main(){*

static int i=i++, j=j++, k=k++;

printf("i = %d j = %d k = %d", i, j, k);

}

Answer:

i = 1 j = 1 k = 1

Explanation:

Since static variables are initialized to zero by default.

98. *void main(){*

```
        while(1){
                if(printf("%d",printf("%d")))
                        break;
                else
                        continue;
        }
}
```

Answer:

Garbage values

Explanation:

The inner printf executes first to print some garbage value. The printf returns no of characters printed and this value also cannot be predicted. Still the outer printf prints something and so returns a non-zero value. So it encounters the break statement and comes out of the while statement.

99.
```
main(){
        unsigned int i=10;
        while(i-->=0)
                printf("%u ",i);
}
```

Answer:

10 9 8 7 6 5 4 3 2 1 0 65535 65534…..

Explanation:

Since i is an unsigned integer it can never become negative. So the expression i-->=0 will always be true, leading to an infinite loop.

100. *main(){*

int x,y=2,z,a;

if(x=y%2) z=2;

a=2;

printf("%d %d ",z,x);

}

Answer:

Garbage-value 0

Explanation:

The value of y%2 is 0. This value is assigned to x. The condition reduces to if (x) or in other words if(0) and so z goes uninitialized.

Thumb Rule: **C**heck all control paths to write bug free code.

101. *main(){*

int a[10];

*printf("%d",*a+1-*a+3);*

}

Answer:

4

Explanation:

*a and -*a cancels out. The result is as simple as 1 + 3 = 4 !

102. *main(){*

unsigned int i=65000;

while(i++!=0);

printf("%d",i);

}

Answer:

1

Explanation:

Note the semicolon after the while statement. When the value of i becomes 0 it comes out of while loop. Due to post-increment on i the value of i while printing is 1.

103. *main(){*

int i=0;

while(+(+i--)!=0)

i-=i++;

printf("%d",i);

}

Answer:

-1

Explanation:

Unary + is the only dummy operator in C. So it has no effect on the expression and now the while loop is, while(i--!=0) which is false and so breaks out of while loop. The value –1 is printed due to the post-decrement operator.

104. *main(){*

```
float f=5,g=10;
enum{i=10,j=20,k=50};
printf("%d\n",++k);
printf("%f\n",f<<2);
printf("%lf\n",f%g);
printf("%lf\n",fmod(f,g));
```

}

Answer:

Line no 5: Error: Lvalue required

Line no 6: Cannot apply leftshift to float

Line no 7: Cannot apply mod to float

Explanation:

Enumeration constants cannot be modified, so you cannot apply ++.Bit-wise operators and % operators cannot be applied on float values.fmod() is to find the modulus values for floats as % operator is for ints.

105. *main(){*

```
int i=10;
```

```
    void pascal f(int,int,int);
    f(i++,i++,i++);
    printf(" %d",i);
}
void pascal f(integer :i,integer:j,integer :k){
    write(i,j,k);
}
```

Answer:

Compiler error: unknown type integer

Compiler error: undeclared function write

Explanation:

Pascal keyword doesn't mean that pascal code can be used. It means that the function follows Pascal argument passing mechanism in calling the functions.

106.
```
void pascal f(int i,int j,int k){
    printf("%d %d %d",i, j, k);
}
void cdecl f(int i,int j,int k){
    printf("%d %d %d",i, j, k);
}
main(){
    int i=10;
    f(i++,i++,i++);
```

```
    printf(" %d\n",i);
    i=10;
    f(i++,i++,i++);
    printf(" %d",i);
}
```

Answer:

10 11 12 13

12 11 10 13

Explanation:

Pascal argument passing mechanism forces the arguments to be called from left to right. cdecl is the normal C argument passing mechanism where the arguments are passed from right to left.

107. *What is the output of the program given below*

```
main(){
    signed char i=0;
    for(;i>=0;i++) ;
    printf("%d\n",i);
}
```

Answer:

-128

Explanation:

Notice the semicolon at the end of the for loop. THe initial value of the i is set to 0. The inner loop executes to increment the value from 0 to 127 (the positive range of char) and then it rotates to the negative value of -128. The condition in the for loop fails and so comes out of the for loop. It prints the current value of i that is -128.

108.
```
main(){
    unsigned char i=0;
    for(;i>=0;i++) ;
        printf("%d\n",i);
}
```

Answer:

infinite loop

Explanation:

The difference between the previous question and this one is that the char is declared to be unsigned. So the i++ can never yield negative value and i>=0 never becomes false so that it can come out of the for loop.

109.
```
main(){
    char i=0;
    for(;i>=0;i++) ;
      printf("%d\n",i);

}
```

Answer:

Behavior is implementation dependent.

Explanation:

The detail if the char is signed/unsigned by default is implementation dependent. If the implementation treats the char to be signed by default the program will print –128 and terminate. On the other hand if it considers char to be unsigned by default, it goes to infinite loop.

Rule:

You can write programs that have implementation dependent behavior. But dont write programs that depend on such behavior.

110. *Is the following statement a declaration/definition. Find what does it mean?*

int (*x)[10];

Answer:

Definition. x is a pointer to array of(size 10) integers.

Apply clock-wise rule to find the meaning of this definition.

111. *What is the output for the program given below?*

```
typedef enum errorType{warning, error, exception,}error;
main() {
    error g1;
    g1=1;
    printf("%d",g1);
```

}

Answer:

Compiler error: Multiple declaration for error

Explanation:

The name error is used in the two meanings. One means that it is a enumerator constant with value 1. The another use is that it is a type name (due to typedef) for enum errorType. Given a situation the compiler cannot distinguish the meaning of error to know in what sense the error is used:

error g1;

g1=error;

// which error it refers in each case?

When the compiler can distinguish between usages then it will not issue error (in pure technical terms, names can only be overloaded in different namespaces).

Note:

The extra comma in the declaration,enum errorType{warning, error, exception,}

is not an error. An extra comma is valid and is provided just for programmer's convenience.

112. *typedef struct error{int warning, error, exception;}error;*

main(){

error g1;

g1.error =1;

printf("%d",g1.error);

}

Answer:

1

Explanation:

The three usages of name errors can be distinguishable by the compiler at any instance, so valid (they are in different namespaces).

Typedef struct error{int warning, error, exception;}error;

This error can be used only by preceding the error by struct kayword as in:

struct error someError;

typedef struct error{int warning, error, exception;}error;

This can be used only after . (dot) or -> (arrow) operator preceded by the variable name as in :

g1.error =1;

printf("%d",g1.error);

typedef struct error{int warning, error, exception;}error;

This can be used to define variables without using the preceding struct keyword as in:

error g1;

Since the compiler can perfectly distinguish between these three usages, it is perfectly legal and valid.

Note:

This code is given here to just explain the concept behind. In real programming don't use such overloading of names. It reduces the readability of the code. Possible doesn't mean that we should use it!

113.
```
#ifdef something
int some=0;
#endif
main(){
    int thing = 0;
    printf("%d %d\n", some ,thing);
}
```

Answer:

Compiler error : undefined symbol some

Explanation:

This is a very simple example for conditional compilation. The name something is not already known to the compiler making the declaration

```
int some = 0;
```

effectively removed from the source code.

114.
```
#if something == 0
int some=0;
#endif
main(){
```

int thing = 0;

printf("%d %d\n", some ,thing);

}

Answer:

0 0

Explanation:

This code is to show that preprocessor expressions are not the same as the ordinary expressions. If a name is not known the preprocessor treats it to be equal to zero.

115. *What is the output for the following program?*

main(){

int arr2D[3][3];

printf("%d\n", ((arr2D== arr2D)&&(* arr2D == arr2D[0])));*

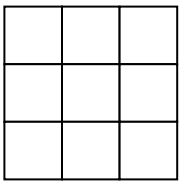
}

Answer:

1

Explanation:

This is due to the close relation between the arrays and pointers. N dimensional arrays are made up of (N-1) dimensional arrays. arr2D is made up of a 3 single arrays that contains 3 integers each .

arr2D  arr2D[1]

arr2D[2]

arr2D[3]

The name arr2D refers to the beginning of all the 3 arrays. *arr2D refers to the start of the first 1D array (of 3 integers) that is the same address as arr2D. So the expression (arr2D == *arr2D) is true (1).

Similarly, *arr2D is nothing but *(arr2D + 0), adding a zero doesn't change the value/meaning. Again arr2D[0] is the another way of telling *(arr2D + 0). So the expression (*(arr2D + 0) == arr2D[0]) is true (1).

Since both parts of the expression evaluates to true the result is true(1) and the same is printed.

116. *void main(){*

if(~0 == (unsigned int)-1)

printf("You can answer this if you know how values are represented in memory");

}

Answer:

You can answer this if you know how values are represented in memory

Explanation:

~ (tilde operator or bit-wise negation operator) operates on 0 to produce all ones to fill the space for an integer. –1 is represented in unsigned value as all 1's and so both are equal.

117. *int swap(int *a,int *b){*

**a=*a+*b;*b=*a-*b;*a=*a-*b;*

}

main() {

int x=10,y=20;

swap(&x,&y);

printf("x= %d y = %d\n",x,y);

}

Answer:

x = 20 y = 10

Explanation:

This is one way of swapping two values. Simple checking will help understand this.

118. main(){

char *p = "ayqm";

printf("%c",++*(p++));

}

Answer:

b

119. *main() {*

int i=5;

printf("%d",++i++);

}

Answer:

Compiler error: Lvalue required in function main

Explanation:

++i yields an rvalue. For postfix ++ to operate an lvalue is required.

120. *main(){*

*char *p = "ayqm";*

char c;

*c = ++*p++;*

printf("%c",c);

}

Answer:

b

Explanation:

There is no difference between the expression ++*(p++) and ++*p++. Parenthesis just works as a visual clue for the reader to see which expression is first evaluated.

121.

```
int aaa() {printf("Hi");}
int bbb(){printf("hello");}
iny ccc(){printf("bye");}
main(){
        int ( * ptr[3]) ();
        ptr[0] = aaa;
        ptr[1] = bbb;
        ptr[2] =ccc;
        ptr[2]();
}
```

Answer:

bye

Explanation:

int (* ptr[3])() says that ptr is an array of pointers to functions that takes no arguments and returns the type int. By the assignment ptr[0] = aaa; it means that the first function pointer in the array is initialized with the address of the function aaa. Similarly, the other two array elements also get initialized with the addresses of the functions bbb and ccc. Since ptr[2] contains the address of the function ccc, the call to the function ptr[2]() is same as calling ccc(). So it results in printing "bye".

122.

```
main(){
        int i=5;
        printf("%d",i=++i ==6);
```

}

Answer:

1

Explanation:

The expression can be treated as i = (++i==6), because == is of higher precedence than = operator. In the inner expression, ++i is equal to 6 yielding true(1). Hence the result.

123. *main(){*

char p[ ]="%d\n";

p[1] = 'c';

printf(p,65);

}

Answer:

A

Explanation:

Due to the assignment p[1] = ‘c’ the string becomes, “%c\n”. Since this string becomes the format string for printf and ASCII value of 65 is ‘A’, the same gets printed.

124. *void (* abc(int, void (*def) ())) ();*

Answer:

abc is a ptr to a function which takes 2 parameters .(. an integer variable.(b). a ptrto a funtion which returns void. the return type of the function is void.

Explanation:

Apply the clock-wise rule to find the result.

125. *main(){*

while (strcmp("some","some\0"))

printf("Strings are not equal\n");

}

Answer:

No output

Explanation:

Ending the string constant with \0 explicitly makes no difference. So "some" and "some\0" are equivalent. So, strcmp returns 0 (false) hence breaking out of the while loop.

126. *main(){*

char str1[] = {'s','o','m','e'};

char str2[] = {'s','o','m','e','\0'};

while (strcmp(str1,str2))

printf("Strings are not equal\n");

}

Answer:

"Strings are not equal"

"Strings are not equal"

….

Explanation:

If a string constant is initialized explicitly with characters, '\0' is not appended automatically to the string. Since str1 doesn't have null termination, it treats whatever the values that are in the following positions as part of the string until it randomly reaches a '\0'. So str1 and str2 are not the same, hence the result.

127. *main(){*

int i = 3;

for (;i++=0;) printf("%d",i);

}

Answer:

Compiler Error: Lvalue required.

Explanation:

As we know that increment operators return rvalues and hence it cannot appear on the left hand side of an assignment operation.

128. *void main(){*

*int *mptr, *cptr;*

mptr = (int)malloc(sizeof(int));*

*printf("%d",*mptr);*

*int *cptr = (int*)calloc(sizeof(int),1);*

*printf("%d",*cptr);*

}

Answer:

garbage-value 0

Explanation:

The memory space allocated by malloc is uninitialized, whereas calloc returns the allocated memory space initialized to zeros.

129. *void main(){*

static int i;

while(i<=10)

(i>2)?i++:i--;

printf("%d", i);

}

Answer:

32767

Explanation:

Since i is static it is initialized to 0. Inside the while loop the conditional operator evaluates to false, executing i--. This continues till the integer value rotates to positive value (32767). The while condition becomes false and hence, comes out of the while loop, printing the i value.

130. *main(){*

int i=10,j=20;

```
    j = i, j?(i,j)?i:j:j;
    printf("%d %d",i,j);
}
```

Answer:

10 10

Explanation:

The Ternary operator (? :) is equivalent for if-then-else statement. So the question can be written as:

```
if(i,j){
    if(i,j)
        j = i;
    else
        j = j;
}
else
    j = j;
```

131. *1. const char *a;*

2. char const a;*

*3. char const *a;*

-Differentiate the above declarations.

Answer:

1. 'const' applies to char * rather than 'a' (pointer to a constant char)

*a='F' : illegal

a="Hi" : legal

2. 'const' applies to 'a' rather than to the value of a (constant pointer to char)

*a='F' : legal

a="Hi" : illegal

3. Same as 1.

132. *main(){*

int i=5,j=10;

i=i&=j&&10;

printf("%d %d",i,j);

}

Answer:

1 10

Explanation:

The expression can be written as i=(i&=(j&&10)); The inner expression (j&&10) evaluates to 1 because j==10. i is 5. i = 5&1 is 1. Hence the result.

133. *main(){*

int i=4,j=7;

j = j || i++ && printf("YOU CAN");

printf("%d %d", i, j);

}

Answer:

4 1

Explanation:

The boolean expression needs to be evaluated only till the truth value of the expression is not known. j is not equal to zero itself means that the expression's truth value is 1. Because it is followed by || and *true || (anything) => true where (anything) will not be evaluated.* So the remaining expression is not evaluated and so the value of i remains the same.

Similarly when && operator is involved in an expression, when any of the operands become false, the whole expression's truth value becomes false and hence the remaining expression will not be evaluated.

false && (anything) => false where (anything) will not be evaluated.

134. main(){

register int a=2;

printf("Address of a = %d",&;

printf("Value of a = %d",;

}

Answer:

Compier Error: '&' on register variable

Rule to Remember:

& (address of) operator cannot be applied on register variables.

135. *main(){*

float i=1.5;

switch(i){

case 1: printf("1");

case 2: printf("2");

default : printf("0");

}

}

Answer:

Compiler Error: switch expression not integral

Explanation:

Switch statements can be applied only to integral types.

136. *main(){*

extern i;

printf("%d\n",i);{

int i=20;

printf("%d\n",i);

}

}

Answer:

Linker Error : Unresolved external symbol i

Explanation:

The identifier i is available in the inner block and so using extern has no use in resolving it.

137. *main(){*

*int a=2,*f1,*f2;*

f1=f2=&a;

**f2+=*f2+=a+=2.5;*

*printf("\n%d %d %d",a,*f1,*f2);*

}

Answer:

16 16 16

Explanation:

f1 and f2 both refer to the same memory location a. So changes through f1 and f2 ultimately affects only the value of a.

138. *main(){*

*char *p="GOOD";*

char a[ ]="GOOD";

*printf("\n sizeof(p) = %d, sizeof(*p) = %d, strlen(p) = %d", sizeof(p), sizeof(*p), strlen(p));*

printf("\n sizeof(= %d, strlen(= %d", sizeof(, strlen();

}

Answer:

sizeof(p) = 2, sizeof(*p) = 1, strlen(p) = 4

sizeof(= 5, strlen(= 4

Explanation:

sizeof(p) => sizeof(char*) => 2

sizeof(*p) => sizeof(char) => 1

Similarly,

sizeof(=> size of the character array => 5

When sizeof operator is applied to an array it returns the sizeof the array and it is not the same as the sizeof the pointer variable. Here the sizeof(where a is the character array and the size of the array is 5 because the space necessary for the terminating NULL character should also be taken into account.

139. *#define DIM(array, type) sizeof(array)/sizeof(type)*

main(){

int arr[10];

printf("The dimension of the array is %d", DIM(arr, int));

}

Answer:

10

Explanation:

The size of integer array of 10 elements is 10 * sizeof(int). The macro expands to sizeof(arr)/sizeof(int) => 10 * sizeof(int) / sizeof(int) => 10.

140. *int DIM(int array[]) {*

return sizeof(array)/sizeof(int);

}

main(){

int arr[10];

printf("The dimension of the array is %d", DIM(arr));

}

Answer:

1

Explanation:

Arrays cannot be passed to functions as arguments and only the pointers can be passed. So the argument is equivalent to int * array (this is one of the very few places where [] and * usage are equivalent). The return statement becomes, sizeof(int *)/ sizeof(int) that happens to be equal in this case.

141. *main(){*

static int a[3][3]={1,2,3,4,5,6,7,8,9};

int i,j;

*static *p[]={a,a+1,a+2};*

for(i=0;i<3;i++){

```
        for(j=0;j<3;j++)
        printf("%d\t%d\t%d\t%d\n",*(*(p+i)+j),
        *(*(j+p)+i),*(*(i+p)+j),*(*(p+j)+i));
    }
}
```

Answer:

1	1	1	1
2	4	2	4
3	7	3	7
4	2	4	2
5	5	5	5
6	8	6	8
7	3	7	3
8	6	8	6
9	9	9	9

Explanation:

((p+i)+j) is equivalent to p[i][j].

142.
```
main(){
    void swap();
    int x=10,y=8;
    swap(&x,&y);
    printf("x=%d y=%d",x,y);
```

}

*void swap(int *a, int *b){*

**a ^= *b, *b ^= *a, *a ^= *b;*

}

Answer:

x=10 y=8

Explanation:

Using ^ like this is a way to swap two variables without using a temporary variable and that too in a single statement.

Inside main(), void swap(); means that swap is a function that may take any number of arguments (not no arguments) and returns nothing. So this doesn't issue a compiler error by the call swap(&x,&y); that has two arguments.

This convention is historically due to pre-ANSI style (referred to as Kernighan and Ritchie style) style of function declaration. In that style, the swap function will be defined as follows,

```
void swap()
int *a, int *b{
        *a ^= *b,  *b ^= *a, *a ^= *b;
        }
```

where the arguments follow the (). So naturally the declaration for swap will look like, void swap() which means the swap can take any number of arguments.

143. *main(){*

```
    int i = 257;
    int *iPtr = &i;
    printf("%d %d", *((char*)iPtr), *((char*)iPtr+1) );
}
```

Answer:

1 1

Explanation:

The integer value 257 is stored in the memory as, 00000001 00000001, so the individual bytes are taken by casting it to char * and get printed.

144.
```
main(){
    int i = 258;
    int *iPtr = &i;
    printf("%d %d", *((char*)iPtr), *((char*)iPtr+1) );
}
```

Answer:

2 1

Explanation:

The integer value 257 can be represented in binary as, 00000001 00000001. Remember that the INTEL machines are ‘small-endian’ machines. *Small-endian means that the lower order bytes are stored in the higher memory addresses and the higher order bytes are stored in lower addresses*. The integer value 258 is stored in memory as: 00000001 00000010.

145. *main(){*

int i=300;

*char *ptr = &i;*

**++ptr=2;*

printf("%d",i);

}

Answer:

556

Explanation:

The integer value 300 in binary notation is: 00000001 00101100. It is stored in memory (small-endian) as: 00101100 00000001. Result of the expression *++ptr = 2 makes the memory representation as: 00101100 00000010. So the integer corresponding to it is 00000010 00101100 => 556.

146. *main()*

{

*char * str = "hello";*

*char * ptr = str;*

char least = 127;

*while (*ptr++)*

*least = (*ptr<least) ?*ptr :least;*

printf("%d",least);

}

Answer:

0

Explanation:

After ‘ptr’ reaches the end of the string the value pointed by ‘str’ is ‘\0’. So the value of ‘str’ is less than that of ‘least’. So the value of ‘least’ finally is 0.

147.

Declare an array of N pointers to functions returning pointers to functions returning pointers to characters?

Answer:

(char*(*)( )) (*ptr[N])( );

148. *main(){*

struct student {

char name[30];

struct date dob;

}stud;

struct date{

int day,month,year;

};

scanf("%s%d%d%d", stud.rollno, &student.dob.day, &student.dob.month, &student.dob.year);

```
}
```

Answer:

Compiler Error: Undefined structure date

Explanation:

Inside the struct definition of ‘student’ the member of type struct date is given. The compiler doesn’t have the definition of date structure (forward reference is not allowed in C in this case) so it issues an error.

149.
```
main(){
    struct date;
    struct student{
        char name[30];
        struct date dob;
    }stud;
    struct date{
        int day,month,year;
     };
    scanf("%s%d%d%d", stud.rollno, &student.dob.day, &student.dob.month,
    &student.dob.year);
}
```

Answer:

Compiler Error: Undefined structure date

Explanation:

Only declaration of struct date is available inside the structure definition of ‘student’ but to have a variable of type struct date the definition of the structure is required.

150.

There were 10 records stored in “somefile.dat” but the following program printed 11 names. What went wrong?

```
void main(){
        struct student{
                char name[30], rollno[6];
        }stud;
        FILE *fp = fopen(“somefile.dat”,”r”);
        while(!feof(fp)) {
                fread(&stud, sizeof(stud), 1 , fp);
                puts(stud.name);
        }
}
```

Explanation:

fread reads 10 records and prints the names successfully. It will return EOF only when fread tries to read another record and fails reading EOF (and returning EOF). So it prints the last record again. After this only the condition feof(fp) becomes false, hence comes out of the while loop.

151. *Is there any difference between the two declarations,*

1. int foo(int *arr[]) and
2. int foo(int *arr[2])

Answer:

No

Explanation:

Functions can only pass pointers and not arrays. The numbers that are allowed inside the [] is just for more readability. So there is no difference between the two declarations.

152. *What is the subtle error in the following code segment?*

```
void fun(int n, int arr[]){
    int *p=0;
    int i=0;
    while(i++<n)
        p = &arr[i];
        *p = 0;
}
```

Answer & Explanation:

If the body of the loop never executes p is assigned no address. So p remains NULL where *p =0 may result in problem (may rise to runtime error “NULL pointer assignment” and terminate the program).

153. *What is wrong with the following code?*

```
int *foo(){
        int *s = malloc(sizeof(int)100);
        assert(s != NULL);
        return s;
}
```

Answer & Explanation:

assert macro should be used for debugging and finding out bugs. The check s != NULL is for error/exception handling and for that assert shouldn't be used. A plain if and the corresponding remedy statement has to be given.

154. *What is the hidden bug with the following statement?*

```
assert(val++ != 0);
```

Answer & Explanation:

Assert macro is used for debugging and removed in release version. In assert, the experssion involves side-effects. So the behavior of the code becomes different in case of debug version and the release version thus leading to a subtle bug.

Rule to Remember:

Don't use expressions that have side-effects in assert statements.

155. *void main(){*

```
        int *i = 0x400;  // i points to the address 400
        *i = 0;          // set the value of memory location pointed by i;
```

}

Answer:

Undefined behavior

Explanation:

The second statement results in undefined behavior because it points to some location whose value may not be available for modification. *This type of pointer in which the non-availability of the implementation of the referenced location is known as 'incomplete type'*.

156. *#define assert(cond) if(!(cond)) *

(fprintf(stderr, "assertion failed: %s, file %s, line %d \n",#cond,

__FILE__,__LINE__), abort())

void main(){

int i = 10;

if(i==0)

assert(i < 100);

else

printf("This statement becomes else for if in assert macro");

}

Answer:

No output

Explanation:

The else part in which the printf is there becomes the else for if in the assert macro. Hence nothing is printed. The solution is to use conditional operator instead of if statement

#define assert(cond) ((cond)?(0): (fprintf (stderr, "assertion failed: \ %s, file %s, line %d \n",#cond, __FILE__,__LINE__), abort()))

Note:

However this problem of "matching with nearest else" cannot be solved by the usual method of placing the if statement inside a block like this,

```
#define assert(cond) { \
if(!(cond)) \
  (fprintf(stderr, "assertion failed: %s, file %s, line %d \n",#cond,\
 __FILE__,__LINE__), abort()) \
}
```

157. *Is the following code legal?*

```
struct a   {
        int x;
         struct a b;
}
```

Answer:

No

Explanation:

Is it not legal for a structure to contain a member that is of the same type as in this case. Because this will cause the structure declaration to be recursive without end.

158. *Is the following code legal?*

```
struct a  {
        int x;
        struct a *b;
}
```

Answer:

Yes.

Explanation:

*b is a pointer to type struct a and so is legal. The compiler knows, the size of the pointer to a structure even before the size of the structure is determined(as you know the pointer to any type is of same size). This type of structures is known as ‘self-referencing’ structure.

159. *Is the following code legal?*

```
typedef struct a {
        int x;
        aType *b;
}aType;
```

Answer:

No

Explanation:

The typename aType is not known at the point of declaring the structure (forward references are not made for typedefs).

160. *Is the following code legal?*

```
typedef struct a aType;
struct a{
    int x;
    aType *b;
};
```

Answer:

Yes

Explanation:

The typename aType is known at the point of declaring the structure, because it is already typedefined.

161. *Is the following code legal?*

```
void main(){
    typedef struct a aType;
    aType someVariable;
    struct a {
        int x;
        aType *b;
```

};

}

Answer:

No

Explanation:

When the declaration,typedef struct a aType; is encountered body of struct a is not known. This is known as 'incomplete types'.

162. *void main(){*

*printf("sizeof (void *) = %d \n", sizeof(void *));*

*printf("sizeof (int *) = %d \n", sizeof(int *));*

*printf("sizeof (double *) = %d \n", sizeof(double *));*

*printf("sizeof(struct unknown *) = %d \n", sizeof(struct unknown *));*

}

Answer:

sizeof (void *) = 2

sizeof (int *) = 2

sizeof (double *) = 2

sizeof(struct unknown *) = 2

Explanation:

The pointer to any type is of same size.

163. *char inputString[100] = {0};*

To get string input from the keyboard which one of the following is better?

1) gets(inputString)

2) fgets(inputString, sizeof(inputString), fp)

Answer & Explanation:

The second one is better because gets(inputString) doesn't know the size of the string passed and so, if a very big input (here, more than 100 chars) the charactes will be written past the input string. When fgets is used with stdin performs the same operation as gets but is safe.

164. *Which version do you prefer of the following two,*

1) printf("%s",str); // or the more curt one

2) printf(str);

Answer & Explanation:

Prefer the first one. If the str contains any format characters like %d then it will result in a subtle bug.

165. *void main(){*

int i=10, j=2;

*int *ip= &i, *jp = &j;*

*int k = *ip/*jp;*

printf("%d",k);

}

Answer:

Compiler Error: “Unexpected end of file in comment started in line 5”.

Explanation:

The programmer intended to divide two integers, but by the “maximum munch” rule, the compiler treats the operator sequence / and * as /* which happens to be the starting of comment. To force what is intended by the programmer,

```
int k = *ip/ *jp;
// give space explicity separating / and *
//or
int k = *ip/(*jp);
// put braces to force the intention
```

will solve the problem.

166. *void main(){*

```
    char ch;
    for(ch=0;ch<=127;ch++)
        printf("%c   %d \n", ch, ch);
}
```

Answer:

Implementaion dependent

Explanation:

The char type may be signed or unsigned by default. If it is signed then ch++ is executed after ch reaches 127 and rotates back to -128. Thus ch is always smaller than 127.

167. *Is this code legal?*

int *ptr;

ptr = (int *) 0x400;***Answer:***

Yes

Explanation:

The pointer ptr will point at the integer in the memory location 0x400.

168. main(){

char a[4]="HELLO";

printf("%s",;

}

Answer:

Compiler error: Too many initializers

Explanation:

The array a is of size 4 but the string constant requires 6 bytes to get stored.

169. main(){

char a[4]="HELL";

printf("%s",;

}

Answer:

HELL%@!~@!@???@~~!

Explanation:

The character array has the memory just enough to hold the string “HELL” and doesnt have enough space to store the terminating null character. So it prints the HELL correctly and continues to print garbage values till it accidentally comes across a NULL character.

170. *main(){*

```
int a=10,*j;
void *k;
j=k=&a;
j++;
k++;
printf("\n %u %u ",j,k);
```

}

Answer:

Compiler error: Cannot increment a void pointer

Explanation:

Void pointers are generic pointers and they can be used only when the type is not known and as an intermediate address storage type. No pointer arithmetic can be done on it and you cannot apply indirection operator (*) on void pointers.

171. *Printf can be implemented by using _____ list.*

Answer:

Variable length argument lists

172. *char *someFun(){*

*char *temp = "string constant";*

return temp;

}

int main(){

puts(someFun());

}

Answer:

string constant

Explanation:

The program suffers no problem and gives the output correctly because the character constants are stored in code/data area and not allocated in stack, so this doesn't lead to dangling pointers.

173. *char *someFun1(){*

char temp[ ] = "string";

return temp;

}

*char *someFun2(){*

char temp[ ] = {'s', 't','r','i','n','g'};

return temp;

}

```
int main(){
        puts(someFun1());
        puts(someFun2());
}
```

Answer:

Garbage values.

Explanation:

Both the functions suffer from the problem of dangling pointers. In someFun1() temp is a character array and so the space for it is allocated in heap and is initialized with character string “string”. This is created dynamically as the function is called, so is also deleted dynamically on exiting the function so the string data is not available in the calling function main() leading to print some garbage values. The function someFun2() also suffers from the same problem but the problem can be easily identified in this case.

174. *Explain Internal linkage.*

Internal linkage means that all declarations of the identifier within one source file refer to a single entity but declarations of the same identifier in other source files refer to different entities.

175. *Can the formal parameter to a function be declared static?*

No, because arguments are always passed on the stack to support recursion.

176. *What is an lvalue?*

Something that can appear on the left side of the "=" sign, it identifies a place where the result can be stored. For example, in the equation a=b+25, a is an lvalue.

In the equation b+25=a, b+25 cannot be used as an lvalue, because it does not identify a specific place. Hence the above assignment is illegal.

177. *Every expression that is an lvalue, is also an rvalue. Is the reverse true?*

No, lvalue denotes a place in the computer's memory. An rvalue denotes a value, so it can only be used on the right hand side of an assignment.

178. *What happens if indirection is performed on a NULL pointer?*

On some machines the indirection accesses the memory location zero. On other machines indirection on a NULL pointer cause a fault that terminate the program. Hence the result is implementation dependent.

179. *Is the statement legal? d=10-*d.*

Illegal because it specifies that an integer quantity (10-*d) be stored in a pointer variable

180. *What does the below indicate?*

*int *func(void)*

a. void indicates that there aren't any arguments.
b. there is one argument of type void.

Answer: a

181. What are data type modifiers?

To extend the data handling power, C adds 4 modifiers which may only be applied to char and int. They are namely signed, unsigned, long and short. Although long may also be applied to double.

182. Interpret the meaning of the following.

a. “ab”,”a+b”

b. “w+t”

Answer:

"ab","a+b"->open a binary file for appending

"w+t" ->create a text file for reading and writing.

183. What is NULL in the context of files?

In using files, if any error occurs, a NULL pointer is returned.

184. What is Register storage class?

It concerns itself with storing data in the registers of the microprocessor and not in memory. The value of the variable doesn't have to be loaded freshly from memory every time. It's important to realize that this a request to the compiler and not a directive. The compiler may not be able to do it. Since the registers are normally of 16 bits long, we can use the register storage class for ints and char's.

185. What is an assertion statement?

They are actually macros. They test statements you pass to them and if the statement is false, they halt the program and inform you that the assertion failed. It is defined in the header file <assert.h>.

*186. Parse int *(*(*(*abc))[6])();*

abc is a pointer to a function returning a pointer to array of pointer to functions returning pointer to integer.

C++ and Java Aptitude

Section I – C++

Note: All the programs are tested under Turbo C++ 3.0/4.5 and Microsoft VC++ 6.0 compilers.

It is assumed that,

- Programs run under DOS/Windows environment,
- Proper and required header files are inlcuded,
- The underlying machine is an x86 based system,

The program output may depend on the information based on this assumptions.

1. *What is the output of the following program?*

```
void main(){
        char str[]  = "String continued"
                " at the next line";
        cout << str;
}
```

Answer:

String continued at the next line.

Explanation:

C++ concatenates these strings. When a new-line or white spaces separate two adjacent strings, they are concatenated to a single string. This operation is called as "stringization" operation. So str could have initialized by the single string "String continued at the next line.".

2. *void main(){*

```
int a, *pa, &ra;
pa = &a;
ra = a;
cout <<"a="<<a <<"*pa="<<*pa <<"ra"<<ra ;
```

}

Answer :

Compiler Error: 'ra', reference must be initialized

Explanation :

Pointers are different from references. One of the main differences is that the pointers can be both initialized and assigned, whereas references can only be initialized. So this code issues an error.

3. *ANSI C++ introduces new style of casts (const_cast, dynamic_cast, static_cast and reinterpret_cast). For the following C style cast which of the new cast syntax should be applied?*

pDerived = (Cderived *) pBase;

Answer:

dynamic_cast.

Explanation:

dynamic_cast is used for traversing up and down in inheritance hierarchy. So it can be applied here to cast pointer type from the base class to derived class. The cast in the new syntax will look like this:

pDerived = dynamic_cast <Cderived*> (pBase);

4. *const int size = 5;*

*void print(int *ptr){*

cout<<ptr[0];

}

void print(int ptr[size]){

cout<<ptr[0];

}

void main(){

int a[size] = {1,2,3,4,5};

*int *b = new int(size);*

print(;

print(b);

}

Answer:

Compiler Error : function 'void print(int *)' already has a body

Explanation:

Arrays cannot be passed to functions, only pointers (for arrays, base addresses) can be passed. So the arguments int *ptr and int prt[size] have no difference as function arguments. In other words, both the functoins have the same signature and so cannot be overloaded.

5. *In which of the following cases will the copy constructor of the class X be called?*

A) X x1,Y Y!:

```
foo(&x1,&y2);          //function  foo already defined
```

B) X x1,x2, Y y1;

```
x2 = bar(x1,y1);      // function bar already defined
```

C) X x1(0.2) ,x2;x2 =x1;

D) X x1; X x2 =x1;

Answer:

B, C and D

Explanation:

In the function call foo, the address of the x1 is passed. So there is no need for the copy constructor. But in the other cases there is an assignment of X, which calls for a copy constructor.

6.

```
class Sample{
public:
    int *ptr;
    Sample(int i){
```

```
        ptr = new int(i);
    }
    ~Sample(){
        delete ptr;
    }
    void PrintVal(){
        cout << "The value is " << *ptr;
    }
};
void SomeFunc(Sample x){
    cout << "Say i am in someFunc " << endl;
}
int main(){
    Sample s1= 10;
    SomeFunc(s1);
    s1.PrintVal();
}
```

Answer:

Say i am in someFunc.

Null pointer assignment(Run-time error)

Explanation:

As the object is passed by value to ‘SomeFunc’ the destructor of the object is called when the control returns from the function. So when PrintVal is called, it meets up

with ptr that has been freed.The solution is to pass the Sample object by *reference* to SomeFunc:

```
void SomeFunc(Sample &x){
        cout << "Say i am in someFunc " << endl;
}
```

because when we pass objects by refernece that object is not destroyed. while returning from the function.

7. *What is the output of the following program?*

```
void Sample(int x=6,int y,int z=34){
        cout << "Say i am in sample with the values " <<x<<y<<z;
}
void main(){
        int a,b,c;
        cin >>a>>b>>c;
        Sample(a,b,c);
}
```

Answer:

Compile Time error: Missing default parameter for parameter 2.

Explanation:

The program wouldn't compile because, as far as default arguments are concerned, the right-most argument must be supplied with a default value before a default argument to a parameter to it's left can be supplied.

8.
```
class base{
        public:
                int bval;
                base(){ bval=0;}
};
class deri:public base{
        public:
                int dval;
                deri(){ dval=1;}
};
void SomeFunc(base *arr,int size){
        for(int i=0; i<size; i++,arr++)
                cout<<arr->bval;
        cout<<endl;
}
int main(){
        base BaseArr[5];
        SomeFunc(BaseArr,5);
        deri DeriArr[5];
        SomeFunc(DeriArr,5);
}
```

Answer:

00000

01010

Explanation:

The function SomeFunc expects two arguments.The first one is a pointer to an array of base class objects and the second one is the sizeof the array.The first call of someFunc calls it with an array of base objects, so it works correctly and prints the bval of all the objects. When SomeFunc is called the second time the argument passed is a pointer to an array of derived class objects and not the array of base class objects. But that is what the function expects to be sent. So the derived class pointer is promoted to base class pointer and the address is sent to the function. SomeFunc() knows nothing about this and just treats the pointer as an array of base class objects. So when arr++ is met, the size of base class object is taken into consideration and is incremented by sizeof(int) bytes for bval (the deri class objects have bval and dval as members and so is of size >= sizeof(int)+sizeof(int)).

9.
```
class some{
    public:
        ~some(){
            cout<<"some's destructor"<<endl;
        }
};
void main(){
    some s;
```

```
    s.~some();
}
```

Answer:

some's destructor

some's destructor

Explanation:

Destructors can be called explicitly (constructors can not be). Here 's.~some()' explicitly calls the destructor of 's'. When main() returns, destructor of s is called again, hence the result.

10.
```
class base{
    public:
        void baseFun(){ cout<<"from base"<<endl;}
};
class deri:public base{
    public:
        void baseFun(){ cout<< "from derived"<<endl;}
};
void SomeFunc(base *baseObj){
    baseObj->baseFun();
}
int main(){
    base baseObject;
```

```
    SomeFunc(&baseObject);
    deri deriObject;
    SomeFunc(&deriObject);
}
```

Answer:

from base

from base

Explanation:

As we have seen in the previous case, SomeFunc expects a pointer to a base class. Since a pointer to a derived class object is passed, it treats the argument only as a base class pointer and the corresponding base function is called.

11.
```
class base{
    public:
        virtual void baseFun(){ cout<<"from base"<<endl;}
};
class deri:public base{
    public:
        void baseFun(){ cout<< "from derived"<<endl;}
};
void SomeFunc(base *baseObj){
    baseObj->baseFun();
}
```

```
int main(){
        base baseObject;
        SomeFunc(&baseObject);
        deri deriObject;
        SomeFunc(&deriObject);
}
```

Answer:

from base

from derived

Explanation:

Remember that baseFunc is a virtual function. That means that it supports run-time polymorphism. So the function corresponding to the derived class object is called.

12. *What is the output of the following code?*

```
class base
{
public:
        int n;
        virtual void foo(){n=1;}
        void print(){cout <<n<<endl;}
};
class derived: base
{
```

```
public:
        void foo(){n=2;}
        void print(){cout <<n<<endl;}
};
void main()
{
        derived y;
        base *bp =dynamic_cast<base *>(&y);
        bp->foo();
        bp->print();
}
```

Answer:

Undefined behavior : dynamic_cast used to convert to inaccessible or ambiguous base;

Explanation:

In this program private inheritance is used (by default the inheritance is private). There is no implicit conversion from the derived to base due to this. An explicit dynamic cast is used to overcome this. But at runtime environment may impose the restrictions on access to the code, resulting in an undefined behavior.

13.
```
class fig2d{
        int dim1, dim2;
        public:
```

```
        fig2d() { dim1=5; dim2=6;}
        virtual void operator<<(ostream & rhs);
};
void fig2d::operator<<(ostream &rhs){
    rhs <<this->dim1<<" "<<this->dim2<<" ";
}
class fig3d : public fig2d{
    int dim3;
    public:
        fig3d() { dim3=7;}
        virtual void operator<<(ostream &rhs);
};
void fig3d::operator<<(ostream &rhs){
    fig2d::operator <<(rhs);
    rhs<<this->dim3;
}
void main(){
    fig2d obj1;
    fig3d obj2;
    obj1 << cout;
    obj2 << cout;
}
```

Answer :

5 6

Explanation:

In this program, the << operator is overloaded with ostream as argument. This enables the 'cout' to be present at the right-hand-side. Normally, operator << is implemented as global function, but it doesn't mean that it is not possible to be overloaded as member function. Overloading << as virtual member function becomes handy when the class in which it is overloaded is inherited, and this becomes available to be overrided. This is as opposed to global friend functions, where friend's are not inherited.

14.
```
class opOverload{
    public:
        bool operator==(opOverload temp);
};
bool opOverload::operator==(opOverload temp){
    if(*this  == temp ){
        cout<<"The both are same objects\n";
        return true;
    }
    cout<<"The both are different\n";
    return false;
}
void main(){
```

opOverload a1, a2;

a1== a2;

}

Answer :

Runtime Error: Stack Overflow

Explanation :

Just like normal functions, operator functions can be called recursively. This program just illustrates that point, by calling the operator == function recursively, leading to an infinite loop.

15. *class complex{*

double re;

double im;

public:

complex() : re(1),im(0.5) {}

operator int(){}

};

int main(){

complex c1;

cout<< c1;

}

Answer :

Garbage value

Explanation:

The programmer wishes to print the complex object using output re-direction operator,which he has not defined for his class.But the compiler instead of giving an error sees the conversion function and converts the user defined object to standard object and prints some garbage value.

16.
```
class complex{
        double re;
        double im;
        public:
                complex() : re(0),im(0) {}
                complex(double n) { re=n,im=n;};
                complex(int m,int n) { re=m,im=n;}
                void print() { cout<<re; cout<<im;}
        };
void main(){
        complex c3;
        double i=5;
        c3 = i;
        c3.print();
}
```

Answer:

5,5

Explanation:

There is no operator= function taking double as an argument is defined. So the compiler creates a 'complex' object using the single argument constructor that takes double as an argument. This temporary object is assigned to c3.

17. *Consider the following piece of code.*

String s("hello!");

String s2 = s1+ "how are You";

Will this code compile for this class declaration:

```
class String {
public:
        String(char *);
        String & operator=(String &);
        String operator+(String);
        friend  operator+(String,String);
};
```

Answer:

No, this code will not compile.

Explanation:

Because there is an ambiguity between the overloading of the + operator as a member function and friend function. Given the statement:

String s2 = s1+ “how are You”;

the compiler doesn’t know to which function should it resolve to and call.

18. *What is the output of the following code?*

```
class base {/*....*/};
class derived :public base{/*....*/};
void foo()
{       try
        {
                throw derived();
        }
        catch (base b)
        {
                cout<<"Received exception, but can't handle\n";
                throw;
        }
};
void main()
{
        try
        {
                foo();
        }
```

```
        catch (derived d)
        {
                cout<<"In derived handler";
        }
        catch (base b)
        {
                cout << "In Base handler";
        }
}
```

Answer:

Received exception, but can't handle

In derived handler

Explanation:

When the function foo is called, the exception object of type derived is thrown. Now the catch block for that ‘derived’ exception object is searched but it is found that there is a catch block for ‘base’ is available. Since exception objects of type ‘base’ can handle all such exceptions in its hierarchy, this exception is caught.

A plain ‘throw’ inside a catch indicates a re-throw of the exception caught. So the ‘derived’ exception object is again thrown and is caught by the catch block in main(). Although both the catch blocks are eligible to handle the exception, this time the derived which is defined first will catch the exception.

19. *What is the output of the following program?*

```
void main(){
    vector<int> vec1,vec2;
    vec1.push_back(12);
    vec1.push_back(13);
    vec2.push_back(12);
    vec2.push_back(13);
    cout << (vec1<vec2);
}
```

Answer:

0

Explanation:

The values contained in both the containers are equal. Therefore the comparison returns 0. Each container supports a set of comparison operators. The comparison is based on a pair-wise comparison of the elements of the two containers. If all the elements are equal and if both the containers contain the same number of elements, the two containers are equal; otherwise, they are unequal. A comparison of the first unequal element determines the less-than or greater-than relationship of the 2 containers.

20. *Consider the following lines of code:*

```
list<int> aList;
list<int>::iterator anIterator(&aList);
```

what can you say about the relationship between 'aList' and 'anIterator'?

Answer:

The class iterator is a friend of class list.

Explanation:

Iterators are always friend functions.

21.

```
#include <list>
#include <algorithm>
#include <iostream>
using namespace std;
void main()
{
list<int> ilist;
list<int>::iterator iiter;
for (int i=0;i<10;i++) ilist.push_back(i);
iiter = find(ilist.begin(),ilist.end(),3);
if ( *(iiter+1) ==4) cout <<"yeah ! found";
}
```

Output:

Compile –time error:

Explanation:

The code won't compile because , iterators associated with list containers do not support the operator + used in the expression (*(iter+1) ==4), because iterators associated with list are of type bi-directional iterator, which doesn't support the + operator.

Exercise:

1) Determine the output of the following 'C++' Codelet.

```
class base{
        public :
        out() {
                cout<<"base ";
        }
};
class deri : public base{
        public : out(){
                cout<<"deri ";
        }
};
void main(){
        deri dp[3];
        base *bp = (base*)dp;
        for (int i=0; i<3;i++)
                (bp++)->out();
}
```

2) Is there anything wrong with this C++ class declaration?

```
class something{
```

```
        char *str;
        public:
            something(){
                st = new char[10];
            }
            ~something(){
                delete str;
             }
    };
```

3) Is there anything wrong with this C++ class declaration?

```
    class temp{
        int value1;
        mutable int value2;
        public:
            void fun(int val) const{
                ((temp*) this)->value1 = 10;
                value2 = 10;
            }
    };
```

22. Name the four parts of a declaration.

- A set of specifiers

- A base type
- A declarator
- An optional initializer

23. Differentiate between declaration and definition in C++.

A declaration introduces a name into the program; a definition provides a unique description of an entity (e.g. type, instance, and function). Declarations can be repeated in a given scope, it introduces a name in a given scope. There must be exactly one definition of every object, function or class used in a C++ program.

A declaration is a definition unless:

- it declares a function without specifying its body,
- it contains an extern specifier and no initializer or function body,
- it is the declaration of a static class data member without a class definition,
- it is a class name definition,
- it is a typedef declaration.

A definition is a declaration unless:

- it defines a static class data member,
- it defines a non-inline member function.

24. Differentiate between initialization and assignment

Initialization is the process that creates the first value of an object. Assignment on the other hand is an operation that usually changes the value of an object. Assignment is

expressed using the token '=' but sometimes initialization can also be indicated with that symbol.

```
int a=12;   // Initialization
a=13;       // Assignment
```

Initialization can also occur without the '=' token, as well. For example,

```
int a(12);
```

For classes, you can define your own initialization procedure by providing constructors and your own assignment operation by providing constructors and your own assignment operation by providing an operator '=' will not be called for initialization is expressed using the '=' symbol.

25. Find five different C++ constructs for which the meaning is undefined an for which the meaning is implementation defined.

Undefined behavior in C++

1. Access outside the bounds of an array.

```
int a[10];
int *p=&a[10];
```

2. Use of a destroyed object

```
int &r=*new int;
delete &r;
r=r+1;
```

3. Attempting to reinterpret variables

```
int i;
```

```
*(float*)&i=1.0;
//Access through undeclared type
```

4. Casting to a type that is not the real type of the source object

```
struct B {int i;}
struct D:B{}
B *bp=new B;
D* dp=static_cast <D*> (bp);   //Invalid cast
```

5. Casting away constness of an object that was defined const

```
void f(int const &r)
{
    const_cast <int&>(r)=2;
}
int const i=3;
f(i);                // Invalid
```

Five implementation-defined constructs in C++

1. Size of certain types:

```
sizeof(int);
```

2. Output of type_info :: name()

```
std::cout<<typeid(double).name()
```

3. The effect of bitwise operators

```
int mistery=~0;
```

4. The extreme values of fundamental types

```
int mx = numeric_limits <int> =max()
```

5. The number of temporary copies

```
struct s
{
    s(s const&)
    {
        std::cout<<"copy\n ";
    }
}
s f( )
{
    s a;
    return a;
}
int main()
{
    f();return 0;
}
```

26. When is a volatile qualifier is required to be used?

In certain cases, you don't want the compiler to produce optimizations. When dealing with variables that are accessible both from an interrupt servicing routine (ISR) and by the regular code, the compiler should not make any assumptions about the value

of a variable from one line of code to the next. Similarly, in a multiprocessing environment, there may be other processors that have access to shared memory variables, thereby making changes to them without warning. In such cases that involve memory mapped hardware devices, you need the volatile qualifier. Volatile qualifier prevents any optimizations done on the names they qualify, making the program possible to be used in such environments.

27. *What is the use of qualifier 'mutable'?*

The mutable qualifier is used to indicate that a particular member of a structure or class can be altered even if a particular structure or class variable is a const.

Example:

```
struct data
{
    char name[30];
    mutable int noOfAccesses;
    .....
};
.....
const data d1{"some string ",0,.....};
strcpy(d1.name,"another string"); // not allowed
d1.noOfAccesses++;                // allowed
```

The const qualifier to d1 prevents a program from changing d1's members, but the mutable qualifier to the accesses member shields accesses from that restriction.

28. *In c++ by default, all functions have external storage class, they can be shared across files. What are the functions to which the above statement proves false.*

Inline functions and functions qualified with the keyword static have internal linkage and are confined to the defining file.

29. *What is the ODR?*

The requirement that global objects and functions either have only one definition or have exactly the same definition many times in a program is called the *one definition rule(*ODR).

30. *When do implicit type conversions occur?*

- In an arithmetic expression of mixed types. These are known as arithmetic conversions.
- In the assignment of an expression of one type to an object of another
- When passing arguments to functions
- Function return types.

31. *List the predefined conversions available in C++.*

C++ supplies certain conversions for all types, and these apply to all classes as well. These include,

- The trivial conversions between a class and a reference to that class.
- The trivial conversion between an array of class objects and a pointer to

that class.

- The trivial conversion from a class instance to a const class object.
- The standard conversion from a pointer to a class object to the type void*.

The compiler uses these conversions implicitly in matching argument and parameter types.

32. Define namespace.

It is a feature in c++ to minimize name collisions in the global name space. This namespace keyword assigns a distinct name to a library that allows other libraries to use the same identifier names without creating any name collisions. Furthermore, the compiler uses the namespace signature for differentiating the definitions.

32. *Is there any space(memory) associated with namespaces?*

No, because namespaces simply deal with names and no space is allocated for a name within a namespace. It is a way by which variables with same name but with different usage are accommodated into a single program.

33. What are using declarations?

It is possible to make a member of a namespace visible within a program. Using Declarations does this. For example:

```
namespace  MySpace{
char * FileName;
char * FileBuffer[300];
```

```
}
using MySpace::FileName;
void main(){
cin >>FileName;
}
```

In this example we have a namespace that consists of two data members. But we have decided to use only one of those data members. We have specified that we want to use only the member FileName.

34. *What are using directives?*

Whenever we want to make all the members of a namespace visible within a given program, we use a using directive. This gives a direction to the compiler that it has to look into this namespace for name resolution.

```
using namespace MySpace;
void main(){
cin >>FileName;
}
```

Here all the data members of the namespace MySpace are visible within the program.

35. When does a name clash occur?

A name clash occurs when a name is defined in more than one place. For example., two different class libraries could give two different classes the same name. If

you try to use many class libraries at the same time, there is a fair chance that you will be unable to compile or link the program because of name clashes.

36. How can a '::' operator be used as unary operator?

The scope operator can be used to refer to members of the global namespace. Because the global namespace doesn't have a name, the notation :: member-name refers to a member of the global namespace. This can be useful for referring to members of global namespace whose names have been hidden by names declared in nested local scope. Unless we specify to the compiler in which namespace to search for a declaration, the compiler simple searches the current scope, and any scopes in which the current scope is nested, to find the declaration for the name.

37. ANSI C++ introduces four version of casts, for four different conversions. What are they and give their purpose.

The new casting syntax introduced are const_cast, static_cast, dynamic_cast and reinterpret_cast.

The general syntax for ANSI C++ style casting is :

toTypeVar = xxx_cast < toType > (fromTypeVar);

This casting syntax helps programmer to identify the casting that are done in program, and its purpose easily than in C, where it can be easily missed.

const_cast is to be used to remove the const or volatileness involved with that object.

char *str = "something";

strlen(const_cast< const char * > str);

// strlen requires const char * as its argument.

It can also be used to remove the volatileness of the object, but that requirement arises very rarely.

static_cast shall be used in all the cases where C casts are normally used.

char *a;

int * b = static_cast < int * > (a);

This converts from char * to int * and this reqires static_cast.

dynamic_cast is used for traversing up and down in inheritance hierarchy.

Base *bp = new Base();

Deri *dp = dynamic_cast < Deri * > (bp);

It should be noted that this is applicable only to 'safe casts', the types that contain the virtual functions.

reinterpret_cast is the cast that can be applied for pointers (particularly function pointers)

int foo();

void (*fp)() = reinterpret_cast< void (*)() > (foo);

fp();

38. When can you tell that a memory leak will occur?

A memory leak occurs when a program loses the ability to free a block of dynamically allocated memory.

39. *What is an activation record?*

The entire storage area of a function is known as the activation record. This is allocated from the program's run-time stack. This contains all the information related with the function, such as the value of the parameters passed, the value of the various local variables etc.

40. Differentiate between a deep copy and a shallow copy?

Deep copy involves using the contents of one object to create another instance of the same class. In a deep copy, the two objects may contain ht same information but the target object will have its own buffers and resources. the destruction of either object will not affect the remaining object. The overloaded assignment operator would create a deep copy of objects.

Shallow copy involves copying the contents of one object into another instance of the same class thus creating a mirror image. Owing to straight copying of references and pointers, the two objects will share the same externally contained contents of the other object to be unpredictable.

Using a copy constructor we simply copy the data values member by member. This method of copying is called shallow copy. If the object is a simple class, comprised of built in types and no pointers this would be acceptable. This function would use the values and the objects and its behavior would not be altered with a shallow copy, only the addresses of pointers that are members are copied and not the value the address is pointing to. The data values of the object would then be inadvertently altered by the

function. When the function goes out of scope, the copy of the object with all its data is popped off the stack.

If the object has any pointers a deep copy needs to be executed. With the deep copy of an object, memory is allocated for the object in free store and the elements pointed to are copied. A deep copy is used for objects that are returned from a function.

41. Which is the parameter that is added implicitly to every non-static member function in a class?

'this' pointer

42. *Give few important properties of 'this' pointer*

A) The 'this' pointer is an implicit pointer used by the system.

B) The 'this' pointer is a constant pointer to an object.

C) The object pointed to by the 'this' pointer can be de-referenced and modified.

43. What is a dangling pointer?

A dangling pointer arises when you use the address of an object after its lifetime is over. This may occur in situations like returning addresses of the automatic variables from a function or using the address of the memory block after it is freed.

44. What is an opaque pointer?

A pointer is said to be opaque if the definition of the type to which it points to is not included in the current translation unit. A translation unit is the result of merging an implementation file with all its headers and header files.

45. What is a smart pointer?

A smart pointer is an object that acts, looks and feels like a normal pointer but offers more functionality. In C++, smart pointers are implemented as *template classes* that encapsulate a pointer and override standard pointer operators. They have a number of advantages over regular pointers. They are guaranteed to be initialized as either null pointers or pointers to a heap object. Indirection through a null pointer is checked. No delete is ever necessary. Objects are automatically freed when the last pointer to them has gone away. One significant problem with these smart pointers is that unlike regular pointers, they don't respect inheritance. Smart pointers are unattractive for polymorphic code. Given below is an example for the implementation of smart pointers.

Example:

```
template <class X> class smart_pointer{
public:
        smart_pointer();                    // makes a null pointer
        smart_pointer(const X& x)           // makes pointer to copy of x
        X& operator *( );
        const X& operator*( ) const;
        X* operator->() const;
        smart_pointer(const smart_pointer <X> &);
        const smart_pointer <X> & operator =(const smart_pointer<X>&);
```

```
        ~smart_pointer();
    private:
        //...
    };
```

This class implement a smart pointer to an object of type X. The object itself is located on the heap. Here is how to use it:

```
smart_pointer <employee> p= employee("Harris",1333);
```

Like other overloaded operators, p will behave like a regular pointer,

```
cout<<*p;
p->raise_salary(0.5);
```

46. How will you decide whether to use pass by reference, by pointer and by value?

The selection of the argument passing depends on the situation.

If a function uses passed data without modifying it,

- If the data object is small, such as a built-in data type or a small structure then pass it by value.
- If the data object is an array, use a pointer because that is the only choice. Make the pointer a pointer to const.
- If the data object is a good-sized structure, use a const pointer or a const reference to increase program efficiency. You save the time and space needed to copy a structure or a class design, make the pointer or reference const.

- If the data object is a class object, use a const reference. The semantics of class design often require using a reference. The standard way to pass class object arguments is by reference.

A function modifies data in the calling function,

- If the data object is a built-in data type, use a pointer. If you spot a code like fixit(&x), where x is an int, its clear that this function intends to modify x.
- If the data object is an array, use the only choice, a pointer.
- If the data object is a structure, use a reference or a pointer.
- If the data object is a class object, use a reference.

47. Describe the main characteristics of static functions.

The main characteristics of static functions include,

- It is without the a this pointer,
- It can't directly access the non-static members of its class
- It can't be declared const, volatile or virtual.
- It doesn't need to be invoked through an object of its class, although for convenience, it may.

48. Will the inline function be compiled as the inline function always? Justify.

An inline function is a request and not a command. Hence it won't be compiled as an inline function always.

Inline-expansion could fail if the inline function contains loops, the address of an inline function is used, or an inline function is called in a complex expression. The rules for inlining are compiler dependent.

49. Define a way other than using the keyword inline to make a function inline.

The function must be defined inside the class.

50. What is name mangling?

Name mangling is the process through which your c++ compilers give each function in your program a unique name. In C++, all programs have at-least a few functions with the same name. Name mangling is a concession to the fact that linker always insists on all function names being unique.

Example:

In general, member names are made unique by concatenating the name of the member with that of the class e.g. given the declaration:

```
class Bar{
public:
        int ival;
        ...
};
```

ival becomes something like:

```
        // a possible member name mangling
        ival__3Bar
```

Consider this derivation:

```
class Foo : public Bar {
public:
    int ival;
    ...
}
```

The internal representation of a Foo object is the concatenation of its base and derived class members.

```
// Pseudo C++ code
// Internal representation of Foo
class Foo{
public:
    int ival__3Bar;
    int ival__3Foo;
    ...
};
```

Unambiguous access of either ival members is achieved through name mangling. Member functions, because they can be overloaded, require an extensive mangling to provide each with a unique name. Here the compiler generates the same name for the two overloaded instances(Their argument lists make their instances unique).

51. Name the operators that cannot be overloaded.

sizeof . .* .-> :: ?:

52. List out the rules for selecting a particular function under function overloading.

- An exact match is better than a trivial adjustment.
- A trivial adjustment (e.g. adding const qualification and/or decaying an array to a pointer to its first element) is preferred over an integral promotion.
- Integral promotion - such as conversion from char to int - beat other standard conversions.
- Standard conversions are better than user defined conversions.
- Matching ellipsis arguments is the worst kind of match.

53. What is the dominance rule?

The dominance rule states that if two classes contain the function searched for, and if one class is derived from the another, the derived class dominates.

54. List out the factors for distinguishing overloaded functions

- The functions must contain a different number of arguments,
- At least one of the arguments must be different.

The return type of a function is not a factor in distinguishing overloaded functions.

55. Define the intersection rule for overloaded operators.

For overloaded functions with more than one parameter, argument matching is more complex. So, the intersection rule was adopted which uses the following procedure:

1. For each argument in the invocation, determine the set of function definitions that contain the best matches for that argument's type.

2. Take the intersection of these sets, and if the result is not a single definition, then the call is ambiguous.

3. The resulting definition must also match at least one argument better than every other definition.

56. List out the argument matching rules for overloaded function selection within the same type.

1. No conversions are necessary. Functions returning no argument conversions are the best candidates.

2. Argument promotions are necessary. One or more arguments are promoted along the path

char -->int--> float-->double-->long-->double

3. Arguments' conversions are necessary. One or more arguments are converted according to standard or user defined conversions.

57. List out the functions that are not (and can't be) inherited in C++.

- Constructors,
- Destructors,
- User defined new operators,
- User defined assignment operators,
- Friend relationships.

58. Declaring a static member function with the const keyword doesn't make sense. Why?

The purpose of the const keyword is to prevent the modification of the values through 'this' and makes the following invisible declaration:

const example *const this;

The static member functions are not passed with an invisible 'this' pointer when they are called. So it doesn't make any sense to declare a static member function as const.

59. Can an operator member function be virtual?

It is possible to declare an operator member function to be virtual, because the mechanics of invoking a member operator function and a regular member function are the same.

60. Can an assignment operator function be declared as friend?

No, an assignment operator can be declared only as a member function, never as a friend.

61. Can a constructor be declared as static?

A static member function can be used without referring to a specific object, using only a class name. They have no 'this' pointer, so they cannot access class member data unless they are passed a 'this' pointer explicitly.

We can't declare constructors to be static. If a constructor was allowed to be static, it could be invoked without a ‘this’ pointer and therefore would not be able to build specific objects from raw storage.

62. *List out the differences between the constructors and destructors*

- destructors can be virtual whereas constructors can’t,
- you cannot pass arguments to destructors,
- there can be one destructor can be declared for a given class,
- constructors cannot be called explicitly like the way a normal function is called using their qualified name whereas a destructor can be.

63. *Describe some unique features of constructors and destructors.*

1. They do not have return value declaration.
2. They cannot be inherited.
3. Their addresses cannot be extracted.

64. *What are the situations under which a copy constructor is used?*

- When a new object is initialized to an object of the same class.
- When an object is passed to a function by value.
- When a function returns an object by value.
- When the compiler generates a temporary object.

65. *What are candidate functions?*

Let us consider that we have a set of overloaded functions. The set of all these functions are known as *candidate functions,* because whenever a overloaded function is met in the program they are the candidates for that particular function invocation. In other words, they are the set of functions considered for a resolving a function call. For example:

```
void f();
void f(int){/*do something*/};
void f(double){ /*do something*/};
void main(){
        f(5.6);
}
```

In this example the functions f() ,f(int) and f(double) are the candidate functions .

66. *What are viable functions?*

Viable functions are the ones that can possibly be resolved into a function call; they are a subset of candidate functions. For example:

```
void f();
void f(short){/*do something*/};
void f(double){ /*do something*/};
void main(){
        f(5.6);// two viable functions f(short) and f(double).
}
```

In this case there are 2 viable functions f(short) and f(double), because we can have a standard conversion by which float can be converted to an short or float getting promoted to type double by promotion. But in this case the value 5.6 is promoted to a value of type double and is passed to that function.

67. *When is the bad_alloc exception thrown?*

We know that 'new' operator is used to allocate space for new objects. If the new operator could not find the space needed to allocate memory for the object, then the allocator throws a *bad_alloc* exception.

68. What is placement new?

When you want to call a constructor directly, you use the placement new. Sometimes you have some raw memory that's already been allocated, and you need to construct an object in the memory you have. Operator new's special version placement new allows you to do it.

```
class Widget{
public :
    Widget(int widgetsize);
    ...
    Widget* Construct_widget_int_buffer(void *buffer,int widgetsize){
        return new(buffer) Widget(widgetsize);
    }
    };
```

This function returns a pointer to a Widget object that's constructed within the buffer passed to the function. Such a function might be useful for applications using shared memory or memory-mapped I/O, because objects in such applications must be placed at specific addresses or in memory allocated by special routines.

69. Which operators are called as insertion and extraction operators? Why?

The C++ operator >> is called the extraction operator, used to extract characters from an input stream and the operator << is called the insertion operator, used to extract text into an output stream.

70. List out the iostream manipulators

Dec	--	Display a numeric values in decimal notation.
endl	--	Output a newline char and flush the stream
ends	--	Output a null character
flush	--	Flush the stream buffer
hex	--	Display numeric values in hexadecimal notation
oct	--	Display numeric values in octal notation
ws	--	Skip over leading white space.

71. What is a parameterized type?

A template is a parameterized construct or type containing generic code that can use or manipulate any type. It is called parameterized because an actual type is a parameter of the code body. Polymorphism may be achieved through parameterized

types. This type of polymorphism is called parameteric polymorphism. Parameteric polymorphism is the mechanism by which the same code is used on different types passed as parameters.

72. List some characteristics of templates that macros do not have.

As opposed to macros, templates:

- have linkage for their instantiations,
- obey scoping rules (template maybe visible only inside a namespace),
- Templates can be overloaded or specialized,
- there can be pointers to function template specializations,
- can be recursive.

73. What is the use of 'autoptr' template?

Each use of '*new*' should be paired with a use of '*delete*'. This can lead to problems if a function in which new is used terminates early through an exception being thrown. Using an autoptr object to keep track of an object created by automates the activation of delete.

Example:

```
void model(string &str)
{
    string *ps=new string(str);
  .....
  if(weird_thing())
```

```
            throw exception();
        str=*ps;
    delete ps;
        return;
}
```

If the exception is thrown, the delete statement isn't reached and there is a memory leak. The auto_ptr template defines a pointer-like object that is intended to be assigned an address obtained by new. When an auto_ptr object expires, its destructor uses delete to free the memory.

To create an auto_ptr object, include the memory header file, which includes the auto_ptr template. The template includes the following:

```
template<class x>class auto_ptr{
public:
    explicit auto_ptr(x *p=0) throw();
.....};
```

Thus asking for an auto_ptr of type x gives you an auto_ptr of type x.

```
auto_ptr<double>pd(new double);   //auto_ptr to double
auto_ptr<string>ps(new string);      //auto_ptr to string
```

74. How would you classify friends to templates.

- Non template friends.
- Bound template friends, meaning that the type of the class determines the type of the friend when a class is instantiated.

- Unbound template friends, meaning that all instantiations of the friend are friends to each instantiation of the class.

75. *Differentiate between a template class and class template.*

Template class:

A generic definition or a parameterized class not instantiated until the client provides the needed information. It's jargon for plain templates.

Class template:

A class template specifies how individual classes can be constructed much like the way a class specifies how individual objects can be constructed. It's jargon for plain classes.

76. *What is meant by template specialization?*

Some cases the general template definition may not fit for a particular data type. For example, let us consider a situation where we are writing a generic function for comparing two values.

Example:

```
template <class T>  T max (T t1,T t2) {
        return T1>T2?T1:T2;
}
```

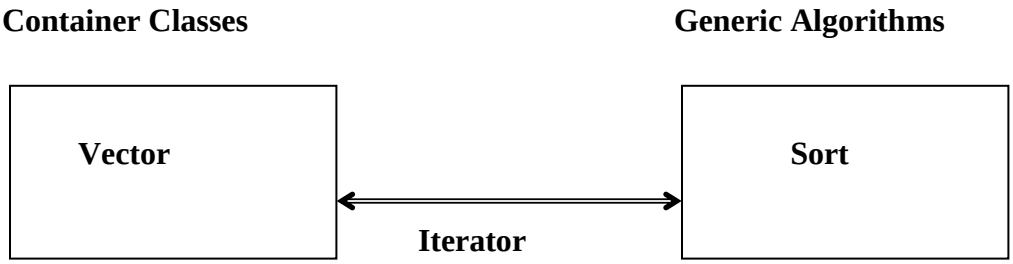
But this template definition won't suit for character strings. So we must have a separate function which implements this function for character strings.

```
template <char *> char * max(char * a,char *b ){
        return ( strcmp(a,b,)>0?a:b);
```

}

77. *What are Iterators in STL?*

Iterators are generalized pointers that may be used to traverse through the contents of a sequence (for example, an STL container or a C++ array). Iterators provide data access for both reading and writing data in a sequence. They serve as intermediaries between containers and generic algorithms. A C++ pointer is an iterator, but an iterator is not necessarily a C++ pointer. Iterators, like pointers can be dereferenced, incremented and decremented. At any point in time, an iterator is positioned at exactly one place in one collection, and remains positioned there until explicitly moved.



78. *How does the allocation of space take place in a vector?*

Initially the size and capacity of a vector are initialized to 0 during declaration. On inserting the first element, the capacity increases to 256, if the vector's elements are of integer type, 128 if they are of type double, 85 if the elements are of type string. As for as classes are concerned, the capacity after initial insertion is 1, because the size and

processing needed for copying or creating a new object of a class is comparatively higher than that for all the above said types. The size of a vector doubles with each reallocation. For example for a vector of type int, during the first reallocation the size allocated is 512 bytes and during the next reallocation it grows to 1024 and so on.

79. What is an Iterator class?

A class that is used to traverse through the objects maintained by a container class. There are five categories of iterators:

- input iterators,
- output iterators,
- forward iterators,
- bidirectional iterators,
- random access.

An iterator is an entity that gives access to the contents of a container object without violating encapsulation constraints. Access to the contents is granted on a one-at-a-time basis in order. The order can be storage order (as in lists and queues) or some arbitrary order (as in array indices) or according to some ordering relation (as in an ordered binary tree). The iterator is a construct, which provides an interface that, when called, yields either the next element in the container, or some value denoting the fact that there are no more elements to examine. Iterators hide the details of access to and update of the elements of a container class.

The simplest and safest iterators are those that permit read-only access to the contents of a container class. The following code fragment shows how an iterator might appear in code:

```
cont_iter:=new cont_iterator();
 x:=cont_iter.next();
while x/=none do
        ...
        s(x);
        ...
    x:=cont_iter.next();
end;
```

In this example, cont_iter is the name of the iterator. It is created on the first line by instantiation of cont_iterator class, an iterator class defined to iterate over some container class, cont. Succesive elements from the container are carried to x. The loop terminates when x is bound to some empty value. (Here, none)In the middle of the loop, there is s(x) an operation on x, the current element from the container. The next element of the container is obtained at the bottom of the loop.

80. What is stack unwinding?

It is a process during exception handling when the destructor is called for all local objects between the place where the exception was thrown and where it is caught. For example:

```
class ExpObj
```

```
{
private:
        int Errno;
public:
        ExpObj(int Eno):Errno(Eno) {};
        void ShowError()
        {
        cout << "The error number is " <<Errno;
        }
};

void sum(int x,int y)
{
        if ( x<0 || y<0)
                throw ExpObj(200);
        int s = x+y;
        cout << "Sum "<<s;
}
void main()
{
        int i=12000, j=-100;
        try
        {
```

```
            sum(i,j);
        }
        catch (ExpObj &Exp)
        {
            Exp.ShowError();
        }
    }
```

In this example, the exception is thrown in the function ‘sum’ and there is neither a try block, nor a catch bolock in the function. The control is shifted to search in the calling function, looking for an exception handler. If one such handler is found, program continues execution. If no such handler is found, the program terminates. This process of searching for exception handler in the calling functions resulting in the discarding of stack frames and subsequent calling of destructors for the local objects is known as ‘stack unwindling’.

Exercise

1. Justify the use of virtual constructors and destructors in C++.
2. Each C++ object possesses the 4 member fns,(which can be declared by the programmer explicitly or by the implementation if they are not available). What are those 4 functions?
3. Inheritance is also known as -------- relationship. Containership as _____ relationship.

4. When is it necessary to use member-wise initialization list (also known as header initialization list) in C++?
5. Which is the only operator in C++ which can be overloaded but NOT inherited.

Section-II Java

Note: All the programs are tested under JDK 1.3 Java compiler.

1.
```
class ArrayCopy{
    public static void main(String[] args){
        int ia1[] = { 1, 2 };
        int ia2[] = (int[])ia1.clone();
        System.out.print((ia1 == ia2) + " ");
        ia1[1]++;
        System.out.println(ia2[1]);
    }
}
```

Answer:

false 2

Explanation:

The clone function creates a new object with a copy of the original object. The == operator compares for checking if the both refer to the same object and returns false (a boolean value) because they are different objects. When concatenated with a string it prints ‘false’ instead of 0.

Since ia1 and ia2 are two different array objects the change in the values stored in ia1 array object doesn’t affect the object ia2.

2. *import javautil.StringTokenizer;*

class STTest {

public static void main(String args[]) {

String s = "9 23 45.4 56.7";

StringTokenizer st = new StringTokenizer(s);

while (st.hasMoreTokens())

System.out.println(st.nextToken());

}

}

Answer: 9

23

45.4

56.7

Explanation:

The StringTokenizer parses the given string to return the individual tokens. Here the String ‘s’ has four white-spaces that act as a separators, resulting in the printing of those individual tokens.

3. *class ConvertTest {*

public static void main (String args[]){

String str;

str = "25";

```
        int i = Integer.valueOf(str).intValue();
        System.out.println(i);
        str = "25.6";
        double d = Double.valueOf(str).doubleValue();
        System.out.println(d);
    }
}
```

Answer:

25 25.6

Explanation:

This program just explains how the static member funntions of the classes Integer and Double can be used to convert the string values that have numbers to the get primitive data-type values.

4.
```
class StaticTest {
    public static void main(String[] args) {
        int i = getX();
    }
    public int getX() {
        return 3;
    }
}
```

Answer:

Compiler Error : Cannot access a non-static member

Explanation:

The static method, main(), belongs to the class. However,getX() belongs to an object in the class. The compiler doesn't know on which object it's invoking the getX() method.There are a couple of ways around this problem. You could declare that getX() as static; that is:

```
public static int getX()
```

Alternately, you can instantiate an object in the StaticTest class in the main() method and invoke that object's getX() method, like this:

```
public static void main(String[] args) {
        StaticTest st = new StaticTest();
        int i = st.getX();
}
```

5.

```
class Test {
    public static void main(String[] args) {
        String s1 = new String("Hello World");
        String s2 = new String("Hello World");
        if (s1 == s2)
            System.out.println("The strings are the same.");
        else
            System.out.println("The strings are different.");
    }
```

```
}
```

Answer:

The strings are different.

Explanation:

When used on objects, == tests whether the two objects are the same object, not whether they have the same value.

To compare two objects for equality, rather than identity, you should use the equals() method.

6.
```
class Test {
    public static void main(String[] args) {
        String s1 = "Hello World";
        String s2 = "Hello World";
        if (s1 == s2)
            System.out.println("The strings are the same");
        else
            System.out.println("The strings are different");
        }
}
```

Answer:

The strings are the same.

Explanation:

Note that these two are string literals and not Strings. The compiler recognizes that the two string literals have the same value and it performs a simple optimization of only creating one String object. Thus s1 and s2 both refer to the same object and are therefore equal. The Java Language Specification requires this behavior. However, not all compilers get this right so in practice this behavior here is implementation dependent.

7.
```
public class works{
    public static int some;
    static {
        some = 100;
        System.out.println("Inside static");
    }
    public static void main( String args[] ) {
        new works();
        System.out.println( "Inside main" );
    }
    works() {
        System.out.println( "some = " + some );
    }
}
```

Answer:

Inside static

some = 100

Inside main

Explanation:

Static blocks are executed before the invocation of main(). So at first the “Inside static” is printed. After that the main function is called. It creates the object of the same type. So it leads to the printing of ‘some = 100’. Finally the println inside the main() is executed to print ‘Inside main’.

8.
```
public class func{
int g(){
        System.out.println("inside g");
        int h(){
                System.out.println("inside h");
                return 1;
        }
        return 0;
}
public static  void  main(String[] args){
        int c;
        c=g();
}
```

Answer:

error : ";" expected at - int h()

Explanation:

Java doesn’t allow function declared within a function declaration (nested functions). Hence the error.

9. *When an exceptional condition causes an exception to be thrown, that exception is an object derived, either directly, or indirectly from the class Exception: True or False?*

Answer:

False.

Explanation:

When an exceptional condition causes an exception to be thrown, that exception is an object derived, either directly, or indirectly from the class Throwable

10. *class Test {*

```
public static void main(String[] args) {
        Button b;
        b.setText("Hello");
        }
}
```

Answer:

Runtime Error : NullPointerException

Explanation:

A NullPointerException is thrown when the system tries to access a object that points to a null value (objects are initalised to null).

This code tries to call setText() on ‘b’. ‘b’ does not refrence any object, so the exception is thrown. You must allocate space for the reference to point to an object like this:

```
Button b = new Button("hello");
        // Or
Button b;
b = new Button();
b.setText("hello");
```

This creates a reference (‘b’) of type Button, then asssigns it to a new instance of Button (new Button("hello")).

11.
```
class Test {
    void Test() {
        System.out.println("Testing") ;
    }
    public static void main(String argv[]) {
       Test ex = new Test() ;
    }
}
```

Answer:

Compiler Error : ‘void’ before Test()

Explanation:

Constructor doesn’t have any return type; so even void shouldn't be specified as return type.

12. *class Test {*

```
int some=10 ;
void Test() {
        this(some++) ;
}
void Test(int i) {
        System.out.println(some);
}
public static void main( String argv[] ) {
        new Test();
 }
}
```

Answer:

Compiler Error : Cannot use ‘this’ inside the constructor

Explanation:

‘this’ is a special one that it refers to the same object. But ‘this’ can be used only after creation of the object. It can’t be used within a constructor. This leads to the issue of the error.

16. *class Test {*

```
    public int some;
    public static void main(String argv[]) {
      int i = new Test().some;
      System.out.println(i) ;
    }
}
```

Answer:

0

Explanation:

All member variables are initialised during creation of the object. In the statement:

int i = new Test().some;

a new object of type ‘Test’ is created and the value of member ‘j’ in that object is referenced (the object created is not assigned to any reference) unig the ‘.’ (dot) operator.

13. *What does this code do?*

```
int f = 1+ (int)(Math.random()*6);
```

This code always assigns an integer to variable f in the range between 1 and 6.

14. *All programs can be written in terms of three types of control structures: What are those three?*

Sequence,selection and repetition.

15. *Is !(x<y) the same as (x>=y) in Java ?*

In the case that x and y are of float or double, the value of x or y could be NaN and the results would be different in that case.

16. *What are peer "classes"?*

Peer classes exist mainly for the convenience of the people who wrote the Java environment. They help in translating between the AWT user interface and the native (Windows, OpenWindows, Mac etc.) interfaces. Unless you're porting Java to a new platform you shouldn't have to use them.

17. *What are concrete classes?*

The classes from which objects are instantiated are called concrete classes (as opposed to abstract classes from which objects cannot be created).

18. *Which methods in Java are implcitly treated as final methods?*

Methods that are declared static and that are declared as private.

19. *Java's finally block provides a mechanism that allows your method to clean up after itself regardless of what happens within the try block. True or False?*

True.

20. *Explain why you should place exception handlers furthermost from the root of the exception hierarchy tree first in the list of exception handlers.*

An exception hander designed to handle a specialized "leaf" object may be preempted by another handler whose exception object type is closer to the root of the exception hierarchy tree if the second exception handler appears earlier in the list of exception handlers.

21. What method of which class would you use to extract the message from an exception object?

The getMessage() method of the Throwable class.

22. What are"deprecated APIs"?

A deprecated API is a object or method that is not recommended to be used and is an indication that it may be removed from the API list in future. The programs that use such APIs still work fine but not recommended to be used for this reason.

UNIX Commands and SQL Statements

Section I – UNIX Commands

1. *What is relative path and absolute path.*

   Absolute path : Exact path from root directory.

   Relative path : Relative to the current path.

2. *Explain kill() and its possible return values.*

   There are four possible results from this call:

   - 'kill()' returns 0. This implies that a process exists with the given PID, and the system would allow you to send signals to it. It is system-dependent whether the process could be a zombie.
   - 'kill()' returns -1, 'errno == ESRCH' either no process exists with the given PID, or security enhancements are causing the system to deny its existence. (On some systems, the process could be a zombie.)
   - 'kill()' returns -1, 'errno == EPERM' the system would not allow you to kill the specified process. This means that either the process exists (again, it could

be a zombie) or draconian security enhancements are present (e.g. your process is not allowed to send signals to *anybody*).

- ‘kill()’ returns -1, with some other value of ‘errno’ you are in trouble! The most-used technique is to assume that success or failure with ‘EPERM’ implies that the process exists, and any other error implies that it doesn't.

An alternative exists, if you are writing specifically for a system (or all those systems) that provide a ‘/proc’ filesystem: checking for the existence of ‘/proc/PID’ may work.

3. *What is a pipe and give an example?*

A pipe is two or more commands separated by pipe char '|'. That tells the shell to arrange for the output of the preceding command to be passed as input to the following command.

Example : ls -l | pr

The output for a command ls is the standard input of pr.

When a sequence of commands are combined using pipe, then it is called pipeline.

4. *How to terminate a process which is running and the specialty on command kill 0?*

With the help of kill command we can terminate the process.

Syntax: kill pid

Kill 0 - kills all processes in your system except the login shell.

5. *What is redirection?*

Directing the flow of data to the file or from the file for input or output.

Example : ls > wc

6. *What are shell variables?*

Shell variables are special variables, a name-value pair created and maintained by the shell.

Example: PATH, HOME, MAIL and TERM

7. *How to switch to a super user status to gain privileges?*

Use ‘su’ command. The system asks for password and when valid entry is made the user gains super user (admin) privileges.

8. *How does the kernel differentiate device files and ordinary files?*

Kernel checks 'type' field in the file's inode structure.

9. *What is the significance of '&' at the end of any command?*

To make the process run as a background process.

10. *How many prompts are available in a UNIX system?*

Two prompts, PS1 (Primary Prompt), PS2 (Secondary Prompt).

11. *Name the data structure used to maintain file identification?*

‘inode’, each file has a separate inode and a unique inode number.

12. *What is the purpose of 'ps' command and its columns?*

- To display process status of all the processes currently running in the system.
- Columns include UID, PID, PPID, STATUS, COMMAND, TTY and TIME.
- Various STATUS type are S - sleeping, R - Running, I - Intermediate.

13. *Is it possible to count number char, line in a file; if so, How?*

Yes, wc-stands for word count.

wc -c for counting number of characters in a file.

wc -l for counting lines in a file.

14. *Is ‘du’ a command? If so, what is its use?*

Yes, it stands for ‘disk usage’. With the help of this command you can find the disk capacity and free space of the disk.

15. *What is the use of the command "ls -x chapter[1-5]"*

ls stands for list; so it displays the list of the files that starts with 'chapter' with suffix '1' to '5', chapter1, chapter2, and so on.

16. *Is it possible to restrict incoming message?*

Yes, using the ‘mesg’ command.

17. *Is it possible to create new a file system in UNIX?*

Yes, ‘mkfs’ is used to create a new file system.

18. *What will the following command do?*

$ echo *

It is similar to 'ls' command and displays all the files in the current directory.

19. *What is the purpose of 'FINGER', 'NICE' and 'TOUCH' command?*

FINGER *:* displays information about all the users logged in.

NICE *:* sets priority of a process range of the NICE value 0 - 39.

TOUCH *:* creates empty files as specified and can also be used to change a file's modified time without modifying the file.

20. *What is the purpose of 'nohup' command?*

A Set of processes are preceded by the command ‘nohup’ along with a output file. Even if the user is logged off the process is not killed and the resulted is evaluated and stored in output file.

21. *Write a command to display a file’s contents in various formats?*

$od -cbd file_name

c - character, b - binary (octal), d-decimal, od=Octal Dump.

22. *Which command is used to delete all files in the current directory and all its sub-directories?*

rm -r *

23. *Write a command to kill the last background job?*

Kill $!

24. *What is the difference between cat and more command*?

Cat displays file contents. If the file is large the contents scroll off the screen before we view it. So command 'more' is like a pager which displays the contents page by page.

25. *What is the use of 'grep' command?*

'grep' is a pattern search command. It searches for the pattern, specified in the command line with appropriate option, in a file(s).

Syntax : *grep* <options> <pattern> <filename(s)>

Example : *grep 99mx mcafile*

26. *What difference between cmp and diff commands?*

cmp - Compares two files byte by byte and displays the first mismatch

diff - tells the changes to be made to make the files identical

27. *Explain the steps that a shell follows while processing a command.*

After the command line is terminated by the <enter> key, the shel goes ahead with processing the command line in one or more passes. The sequence is well defined and assumes the following order.

Parsing: The shell first breaks up the command line into words, using spaces and the delimiters, unless quoted. All consecutive occurrences of a space or tab are replaced here with a single space.

Variable evaluation: All words preceded by a $ are avaluated as variables, unless quoted or escaped.

Command substitution: Any command surrounded by backquotes is executed by the shell which then replaces the standard output of the command into the command line.

Wild-card interpretation: The shell finally scans the command line for wild-cards (the characters *, ?, [,]). Any word containing a wild-card is replaced by a sorted list of filenames that match the pattern. The list of these filenames then forms the arguments to the command.

PATH evaluation: It finally looks for the PATH variable to determine the sequence of directories it has to search in order to hunt for the command.

28. *What is the difference between > and >> redirection operators ?*

> is the output redirection operator when used it overwrites while >> operator appends into the file.

29. *How is the command "$cat <file1 >file2 " different from "$cat >file2 <file1" ?*

Both the commands mean the same , the file file1 is read and its contents are copied to file2

30. *Which of the following commands is not a filter*

(a)man , (b) cat , (c) pg , (d) head

Ans: man

A filter is a program which can receive a flow of data from std input, process (or filter) it and send the result to the std output.

31. *What does the command "$ls | wc –l > file1" do?*

ls becomes the input to wc which counts the number of lines it receives as input and instead of displaying this count , the value is stored in file1.

32. *What does the command " $who | sort –logfile > newfile" do?*

The input from a pipe can be combined with the input from a file . The trick is to use the special symbol "-" (a hyphen) for those commands that recognize the hyphen as std input.

In the above command the output from who becomes the std input to sort , meanwhile sort opens the file logfile, the contents of this file is sorted together with the output of who (rep by the hyphen) and the sorted output is redirected to the file newfile.

33. *What is the significance of the "tee" command?*

It reads the standard input and sends it to the standard output while redirecting a copy of what it has read to the file specified by the user.

34. *Explain the command "$who | tee file1 file2 /dev/tty3a | sort > file3"*

Store the output of who in file1, file2 ,display the same output on the screen store the sorted output int file3. /dev/tty3a is the file associated with the terminal.

Exercise :-

35. *Explain the following commands.*

$ ls > file1

$ banner hi-fi > message

$ cat par.3 par.4 par.5 >> report

$ cat file1>file1

$ date ; who

$ date ; who > logfile

$ (date ; who) > logfile

36. *Construct pipes to execute the following jobs.*

1. Output of who should be displayed on the screen with value of total number of users who have logged in displayed at the bottom of the list.
2. Output of ls should be displayed on the screen and from this output the lines containing the word 'poem' should be counted and the count should be stored in a file.

3. Contents of file1 and file2 should be displayed on the screen and this output should be appended in a file .

4. From output of ls the lines containing ‘poem’ should be displayed on the screen along with the count.

5. Name of cities should be accepted from the keyboard . This list should be combined with the list present in a file. This combined list should be sorted and the sorted list should be stored in a file ‘newcity’.

6. All files present in a directory dir1 should be deleted any error while deleting should be stored in a file ‘errorlog’.

Section II – SQL

1. *Which is the subset of SQL commands used to manipulate Oracle Database structures, including tables?*

   Data Definition Language (DDL)

2. *What operator performs pattern matching?*

   LIKE operator

3. *What operator tests column for the absence of data?*

IS NULL operator

4. *Which command executes the contents of a specified file?*

START <filename> or @<filename>

5. *What is the parameter substitution symbol used with INSERT INTO command?*

&

6. *Which command displays the SQL command in the SQL buffer, and then executes it?*

RUN

7. *What are the wildcards used for pattern matching?*

_ for single character substitution and % for multi-character substitution

8. *State true or false. EXISTS, SOME, ANY are operators in SQL.*

True

9. *State true or false. !=, <>, ^= all denote the same operation.*

True

10. *What are the privileges that can be granted on a table by a user to others?*

Insert, update, delete, select, references, index, execute, alter, all

11. What command is used to get back the privileges offered by the GRANT command?

REVOKE

12. Which system tables contain information on privileges granted and privileges obtained?

USER_TAB_PRIVS_MADE, USER_TAB_PRIVS_RECD

13. Which system table contains information on constraints on all the tables created?

USER_CONSTRAINTS

14. TRUNCATE TABLE EMP;

DELETE FROM EMP;

Will the outputs of the above two commands differ?

Both will result in deleting all the rows in the table EMP.

15. What is the difference between TRUNCATE and DELETE commands?

TRUNCATE is a DDL command whereas DELETE is a DML command. Hence DELETE operation can be rolled back, but TRUNCATE operation cannot be rolled back. WHERE clause can be used with DELETE and not with TRUNCATE.

16. What command is used to create a table by copying the structure of another table?

Answer :

CREATE TABLE .. AS SELECT command

Explanation :

To copy only the structure, the WHERE clause of the SELECT command should contain a FALSE statement as in the following.

CREATE TABLE NEWTABLE AS SELECT * FROM EXISTINGTABLE WHERE 1=2;

If the WHERE condition is true, then all the rows or rows satisfying the condition will be copied to the new table.

17. *What will be the output of the following query?*

*SELECT REPLACE(TRANSLATE(LTRIM(RTRIM('!! ATHEN !!','!'), '!'), 'AN', '**'),'*','TROUBLE') FROM DUAL;*

TROUBLETHETROUBLE

18. *What will be the output of the following query?*

SELECT DECODE(TRANSLATE('A','1234567890','1111111111'), '1','YES', 'NO');

Answer :

NO

Explanation :

The query checks whether a given string is a numerical digit.

19. *What does the following query do?*

SELECT SAL + NVL(COMM,0) FROM EMP;

This displays the total salary of all employees. The null values in the commission column will be replaced by 0 and added to salary.

20. Which date function is used to find the difference between two dates?

MONTHS_BETWEEN

21. Why does the following command give a compilation error?

DROP TABLE &TABLE_NAME;

Variable names should start with an alphabet. Here the table name starts with an '&' symbol.

22. What is the advantage of specifying WITH GRANT OPTION in the GRANT command?

The privilege receiver can further grant the privileges he/she has obtained from the owner to any other user.

23. What is the use of the DROP option in the ALTER TABLE command?

It is used to drop constraints specified on the table.

24. What is the value of 'comm' and 'sal' after executing the following query if the initial value of 'sal' is 10000?

*UPDATE EMP SET SAL = SAL + 1000, COMM = SAL*0.1;*

sal = 11000, comm = 1000

25. What is the use of DESC in SQL?

Answer :

DESC has two purposes. It is used to describe a schema as well as to retrieve rows from table in descending order.

Explanation :

The query SELECT * FROM EMP ORDER BY ENAME DESC will display the output sorted on ENAME in descending order.

26. What is the use of CASCADE CONSTRAINTS?

When this clause is used with the DROP command, a parent table can be dropped even when a child table exists.

27. Which function is used to find the largest integer less than or equal to a specific value?

FLOOR

28. What is the output of the following query?

SELECT TRUNC(1234.5678,-2) FROM DUAL;

1200

SQL – QUERIES

I. SCHEMAS

Table 1 : ***STUDIES***

PNAME (VARCHAR), SPLACE (VARCHAR), COURSE (VARCHAR), CCOST (NUMBER)

Table 2 : ***SOFTWARE***

PNAME (VARCHAR), TITLE (VARCHAR), DEVIN (VARCHAR), SCOST (NUMBER), DCOST (NUMBER), SOLD (NUMBER)

Table 3 : ***PROGRAMMER***

PNAME (VARCHAR), DOB (DATE), DOJ (DATE), SEX (CHAR), PROF1 (VARCHAR), PROF2 (VARCHAR), SAL (NUMBER)

LEGEND :

PNAME – Programmer Name, SPLACE – Study Place, CCOST – Course Cost, DEVIN – Developed in, SCOST – Software Cost, DCOST – Development Cost, PROF1 – Proficiency 1

QUERIES :

1. *Find out the selling cost average for packages developed in Oracle.*
2. *Display the names, ages and experience of all programmers.*
3. *Display the names of those who have done the PGDCA course.*
4. *What is the highest number of copies sold by a package?*
5. *Display the names and date of birth of all programmers born in April.*

6. *Display the lowest course fee.*
7. *How many programmers have done the DCA course.*
8. *How much revenue has been earned through the sale of packages developed in C.*
9. *Display the details of software developed by Rakesh.*
10. *How many programmers studied at Pentafour.*
11. *Display the details of packages whose sales crossed the 5000 mark.*
12. *Find out the number of copies which should be sold in order to recover the development cost of each package.*
13. *Display the details of packages for which the development cost has been recovered.*
14. *What is the price of costliest software developed in VB?*
15. *How many packages were developed in Oracle ?*
16. *How many programmers studied at PRAGATHI?*
17. *How many programmers paid 10000 to 15000 for the course?*
18. *What is the average course fee?*
19. *Display the details of programmers knowing C.*
20. *How many programmers know either C or Pascal?*
21. *How many programmers don't know C and C++?*
22. *How old is the oldest male programmer?*
23. *What is the average age of female programmers?*
24. *Calculate the experience in years for each programmer and display along with their names in descending order.*
25. *Who are the programmers who celebrate their birthdays during the current month?*
26. *How many female programmers are there?*

27. What are the languages known by the male programmers?

28. What is the average salary?

29. How many people draw 5000 to 7500?

30. Display the details of those who don't know C, C++ or Pascal.

31. Display the costliest package developed by each programmer.

32. Produce the following output for all the male programmers

Programmer

Mr. Arvind – has 15 years of experience

KEYS:

1. SELECT AVG(SCOST) FROM SOFTWARE WHERE DEVIN = 'ORACLE';
2. SELECT PNAME,TRUNC(MONTHS_BETWEEN(SYSDATE,DOB)/12) "AGE", TRUNC(MONTHS_BETWEEN(SYSDATE,DOJ)/12) "EXPERIENCE" FROM PROGRAMMER;
3. SELECT PNAME FROM STUDIES WHERE COURSE = 'PGDCA';
4. SELECT MAX(SOLD) FROM SOFTWARE;
5. SELECT PNAME, DOB FROM PROGRAMMER WHERE DOB LIKE '%APR%';
6. SELECT MIN(CCOST) FROM STUDIES;
7. SELECT COUNT(*) FROM STUDIES WHERE COURSE = 'DCA';
8. SELECT SUM(SCOST*SOLD-DCOST) FROM SOFTWARE GROUP BY DEVIN HAVING DEVIN = 'C';
9. SELECT * FROM SOFTWARE WHERE PNAME = 'RAKESH';
10. SELECT * FROM STUDIES WHERE SPLACE = 'PENTAFOUR';

11. SELECT * FROM SOFTWARE WHERE SCOST*SOLD-DCOST > 5000;
12. SELECT CEIL(DCOST/SCOST) FROM SOFTWARE;
13. SELECT * FROM SOFTWARE WHERE SCOST*SOLD >= DCOST;
14. SELECT MAX(SCOST) FROM SOFTWARE GROUP BY DEVIN HAVING DEVIN = 'VB';
15. SELECT COUNT(*) FROM SOFTWARE WHERE DEVIN = 'ORACLE';
16. SELECT COUNT(*) FROM STUDIES WHERE SPLACE = 'PRAGATHI';
17. SELECT COUNT(*) FROM STUDIES WHERE CCOST BETWEEN 10000 AND 15000;
18. SELECT AVG(CCOST) FROM STUDIES;
19. SELECT * FROM PROGRAMMER WHERE PROF1 = 'C' OR PROF2 = 'C';
20. SELECT * FROM PROGRAMMER WHERE PROF1 IN ('C','PASCAL') OR PROF2 IN ('C','PASCAL');
21. SELECT * FROM PROGRAMMER WHERE PROF1 NOT IN ('C','C++') AND PROF2 NOT IN ('C','C++');
22. SELECT TRUNC(MAX(MONTHS_BETWEEN(SYSDATE,DOB)/12)) FROM PROGRAMMER WHERE SEX = 'M';
23. SELECT TRUNC(AVG(MONTHS_BETWEEN(SYSDATE,DOB)/12)) FROM PROGRAMMER WHERE SEX = 'F';
24. SELECT PNAME, TRUNC(MONTHS_BETWEEN(SYSDATE,DOJ)/12) FROM PROGRAMMER ORDER BY PNAME DESC;
25. SELECT PNAME FROM PROGRAMMER WHERE TO_CHAR(DOB,'MON') = TO_CHAR(SYSDATE,'MON');

26. SELECT COUNT(*) FROM PROGRAMMER WHERE SEX = 'F';
27. SELECT DISTINCT(PROF1) FROM PROGRAMMER WHERE SEX = 'M';
28. SELECT AVG(SAL) FROM PROGRAMMER;
29. SELECT COUNT(*) FROM PROGRAMMER WHERE SAL BETWEEN 5000 AND 7500;
30. SELECT * FROM PROGRAMMER WHERE PROF1 NOT IN ('C','C++','PASCAL') AND PROF2 NOT IN ('C','C++','PASCAL');
31. SELECT PNAME,TITLE,SCOST FROM SOFTWARE WHERE SCOST IN (SELECT MAX(SCOST) FROM SOFTWARE GROUP BY PNAME);
32. SELECT 'Mr.' || PNAME || ' - has ' || TRUNC(MONTHS_BETWEEN(SYSDATE,DOJ)/12) || ' years of experience' "Programmer" FROM PROGRAMMER WHERE SEX = 'M' UNION SELECT 'Ms.' || PNAME || ' - has ' || TRUNC (MONTHS_BETWEEN (SYSDATE,DOJ)/12) || ' years of experience' "Programmer" FROM PROGRAMMER WHERE SEX = 'F';

II . SCHEMA :

Table 1 : **DEPT**

DEPTNO (NOT NULL , NUMBER(2)), DNAME (VARCHAR2(14)),

LOC (VARCHAR2(13)

Table 2 : **EMP**

EMPNO (NOT NULL , NUMBER(4)), ENAME (VARCHAR2(10)),

JOB (VARCHAR2(9)), MGR (NUMBER(4)), HIREDATE (DATE),

SAL (NUMBER(7,2)), COMM (NUMBER(7,2)), DEPTNO (NUMBER(2))

MGR is the empno of the employee whom the employee reports to. DEPTNO is a foreign key.

QUERIES

1. *List all the employees who have at least one person reporting to them.*
2. *List the employee details if and only if more than 10 employees are present in department no 10.*
3. *List the name of the employees with their immediate higher authority.*
4. *List all the employees who do not manage any one.*
5. *List the employee details whose salary is greater than the lowest salary of an employee belonging to deptno 20.*
6. *List the details of the employee earning more than the highest paid manager.*
7. *List the highest salary paid for each job.*
8. *Find the most recently hired employee in each department.*
9. *In which year did most people join the company? Display the year and the number of employees.*
10. *Which department has the highest annual remuneration bill?*
11. *Write a query to display a '*' against the row of the most recently hired employee.*
12. *Write a correlated sub-query to list out the employees who earn more than the average salary of their department.*
13. *Find the nth maximum salary.*

14. Select the duplicate records (Records, which are inserted, that already exist) in the EMP table.

15. Write a query to list the length of service of the employees (of the form n years and m months).

KEYS:

1. SELECT DISTINCT(A.ENAME) FROM EMP A, EMP B WHERE A.EMPNO = B.MGR;

or SELECT ENAME FROM EMP WHERE EMPNO IN (SELECT MGR FROM EMP);

2. SELECT * FROM EMP WHERE DEPTNO IN (SELECT DEPTNO FROM EMP GROUP BY DEPTNO HAVING COUNT(EMPNO)>10 AND DEPTNO=10);

3. SELECT A.ENAME "EMPLOYEE", B.ENAME "REPORTS TO" FROM EMP A, EMP B WHERE A.MGR=B.EMPNO;

4. SELECT * FROM EMP WHERE EMPNO IN (SELECT EMPNO FROM EMP MINUS SELECT MGR FROM EMP);

5. SELECT * FROM EMP WHERE SAL > (SELECT MIN(SAL) FROM EMP GROUP BY DEPTNO HAVING DEPTNO=20);

6. SELECT * FROM EMP WHERE SAL > (SELECT MAX(SAL) FROM EMP GROUP BY JOB HAVING JOB = 'MANAGER');

7. SELECT JOB, MAX(SAL) FROM EMP GROUP BY JOB;

8. SELECT * FROM EMP WHERE (DEPTNO, HIREDATE) IN (SELECT DEPTNO, MAX(HIREDATE) FROM EMP GROUP BY DEPTNO);

9. SELECT TO_CHAR(HIREDATE,'YYYY') "YEAR", COUNT(EMPNO) "NO. OF EMPLOYEES" FROM EMP GROUP BY TO_CHAR(HIREDATE,'YYYY') HAVING

COUNT(EMPNO) = (SELECT MAX(COUNT(EMPNO)) FROM EMP GROUP BY TO_CHAR(HIREDATE,'YYYY'));

10. SELECT DEPTNO, LPAD(SUM(12*(SAL+NVL(COMM,0))),15) "COMPENSATION" FROM EMP GROUP BY DEPTNO HAVING SUM(12*(SAL+NVL(COMM,0))) = (SELECT MAX(SUM(12*(SAL+NVL(COMM,0)))) FROM EMP GROUP BY DEPTNO);

11. SELECT ENAME, HIREDATE, LPAD('*',8) "RECENTLY HIRED" FROM EMP WHERE HIREDATE = (SELECT MAX(HIREDATE) FROM EMP) UNION SELECT ENAME NAME, HIREDATE, LPAD(' ',15) "RECENTLY HIRED" FROM EMP WHERE HIREDATE != (SELECT MAX(HIREDATE) FROM EMP);

12. SELECT ENAME,SAL FROM EMP E WHERE SAL > (SELECT AVG(SAL) FROM EMP F WHERE E.DEPTNO = F.DEPTNO);

13. SELECT ENAME, SAL FROM EMP A WHERE &N = (SELECT COUNT (DISTINCT(SAL)) FROM EMP B WHERE A.SAL<=B.SAL);

14. SELECT * FROM EMP A WHERE A.EMPNO IN (SELECT EMPNO FROM EMP GROUP BY EMPNO HAVING COUNT(EMPNO)>1) AND A.ROWID!=MIN (ROWID));

15. SELECT ENAME "EMPLOYEE",TO_CHAR(TRUNC(MONTHS_BETWEEN(SYSDATE,HIREDATE)/12))||' YEARS '|| TO_CHAR(TRUNC(MOD(MONTHS_BETWEEN (SYSDATE, HIREDATE),12)))||' MONTHS ' "LENGTH OF SERVICE" FROM EMP;

Quantitative Aptitude

FORMULA LIST:

ALGEBRA :

1. Sum of first n natural numbers = $n(n+1)/2$
2. Sum of the squares of first n natural numbers = $n(n+1)(2n+1)/6$
3. Sum of the cubes of first n natural numbers = $[n(n+1)/2]^2$
4. Sum of first n natural odd numbers = n^2
5. Average = *(Sum of items)/Number of items*

Arithmetic Progression (A.P.):

An A.P. is of the form a, a+d, a+2d, a+3d, ...

where a is called the 'first term' and d is called the 'common difference'

1. nth term of an A.P. $tn = a + (n-1)d$
2. Sum of the first n terms of an A.P. $Sn = n/2[2a+(n-1)d]$ or *Sn = n/2(first term + last term)*

Geometrical Progression (G.P.):

A G.P. is of the form a, ar, ar^2, ar^3, ...

where a is called the 'first term' and r is called the 'common ratio'.

1. nth term of a G.P. $t_n = ar^{n-1}$
2. Sum of the first n terms in a G.P. $Sn = a|1-r^n|/|1-r|$

Permutations and Combinations :

1. nPr = n!/(n-r)!
2. nPn = n!
3. nP1 = n

1. nCr = n!/(r! (n-r)!)
2. nC1 = n
3. nC0 = 1 = nCn
4. nCr = nCn-r
5. nCr = nPr/r!

Number of diagonals in a geometric figure of n sides = nC2-n

Tests of Divisibility :

1. A number is divisible by 2 if it is an even number.
2. A number is divisible by 3 if the sum of the digits is divisible by 3.
3. A number is divisible by 4 if the number formed by the last two digits is divisible by 4.
4. A number is divisible by 5 if the units digit is either 5 or 0.
5. A number is divisible by 6 if the number is divisible by both 2 and 3.
6. A number is divisible by 8 if the number formed by the last three digits is divisible by 8.

7. A number is divisible by 9 if the sum of the digits is divisible by 9.
8. A number is divisible by 10 if the units digit is 0.
9. A number is divisible by 11 if the difference of the sum of its digits at odd places and the sum of its digits at even places, is divisible by 11.

H.C.F and L.C.M :

H.C.F stands for Highest Common Factor. The other names for H.C.F are Greatest Common Divisor (G.C.D) and Greatest Common Measure (G.C.M).

The H.C.F. of two or more numbers is the greatest number that divides each one of them exactly.

The least number which is exactly divisible by each one of the given numbers is called their L.C.M.

Two numbers are said to be co-prime if their H.C.F. is 1.

H.C.F. of fractions = H.C.F. of numerators/L.C.M of denominators

L.C.M. of fractions = G.C.D. of numerators/H.C.F of denominators

Product of two numbers = Product of their H.C.F. and L.C.M.

PERCENTAGES :

1. If A is R% more than B, then B is less than A by R / (100+R) * 100
2. If A is R% less than B, then B is more than A by R / (100-R) * 100
3. If the price of a commodity increases by R%, then reduction in consumption, not to increase the expenditure is : R/(100+R)*100

4. If the price of a commodity decreases by R%, then the increase in consumption, not to decrease the expenditure is : R/(100-R)*100

PROFIT & LOSS :

1. Gain = Selling Price(S.P.) - Cost Price(C.P)
2. Loss = C.P. - S.P.
3. Gain % = Gain * 100 / C.P.
4. Loss % = Loss * 100 / C.P.
5. S.P. = (100+Gain%)/100*C.P.
6. S.P. = (100-Loss%)/100*C.P.

Short cut Methods:

1. By selling an article for Rs. X, a man loses l%. At what price should he sell it to gain y%?

   (or)

   A man lost l% by selling an article for Rs. X. What percent shall he gain or lose by selling it for Rs. Y?

   (100 – loss%) : 1^{st} S.P. = (100 + gain%) : 2^{nd} S.P.

2. A man sold two articles for Rs. X each. On one he gains y% while on the other he loses y%. How much does he gain or lose in the whole transaction?

   In such a question, there is always a lose. The selling price is immaterial.

$$\text{Formula: Loss \%} = \left(\frac{\text{Common loss or gain\%}}{10} \right)^2 \%$$

3. A discount dealer professes to sell his goods at cost price but uses a weight of 960 gms. For a kg weight. Find his gain percent.

$$\text{Formula: Gain \%} = \left(\frac{\text{Error}}{\text{True value - Error}} * 100 \right) \%$$

RATIO & PROPORTIONS:

1. The ratio a : b represents a fraction a/b. a is called antecedent and b is called consequent.
2. The equality of two different ratios is called proportion.
3. If a : b = c : d then a, b, c, d are in proportion. This is represented by a : b :: c : d.
4. In a : b = c : d, then we have a* d = b * c.
5. If a/b = c/d then (a + b) / (a – b) = (d + c) / (d – c).

TIME & WORK :

1. If A can do a piece of work in n days, then A's 1 day's work = 1/n
2. If A and B work together for n days, then (A+B)'s 1 days's work = 1/n
3. If A is twice as good workman as B, then ratio of work done by A and B = 2:1

PIPES & CISTERNS :

1. If a pipe can fill a tank in x hours, then part of tank filled in one hour = 1/x
2. If a pipe can empty a full tank in y hours, then part emptied in one hour = 1/y
3. If a pipe can fill a tank in x hours, and another pipe can empty the full tank in y hours, then on opening both the pipes,

   the net part filled in 1 hour = (1/x-1/y) if y>x

   the net part emptied in 1 hour = (1/y-1/x) if x>y

TIME & DISTANCE :

1. Distance = Speed * Time
2. 1 km/hr = 5/18 m/sec
3. 1 m/sec = 18/5 km/hr
4. Suppose a man covers a certain distance at x kmph and an equal distance at y kmph. Then, the average speed during the whole journey is 2xy/(x+y) kmph.

PROBLEMS ON TRAINS :

1. Time taken by a train x metres long in passing a signal post or a pole or a standing man is equal to the time taken by the train to cover x metres.
2. Time taken by a train x metres long in passing a stationary object of length y metres is equal to the time taken by the train to cover x+y metres.

3. Suppose two trains are moving in the same direction at u kmph and v kmph such that u>v, then their relative speed = u-v kmph.
4. If two trains of length x km and y km are moving in the same direction at u kmph and v kmph, where u>v, then time taken by the faster train to cross the slower train = (x+y)/(u-v) hours.
5. Suppose two trains are moving in opposite directions at u kmph and v kmph. Then, their relative speed = (u+v) kmph.
6. If two trains of length x km and y km are moving in the opposite directions at u kmph and v kmph, then time taken by the trains to cross each other = (x+y)/(u+v)hours.
7. If two trains start at the same time from two points A and B towards each other and after crossing they take a and b hours in reaching B and A respectively, then A's speed : B's speed = (√b : √

SIMPLE & COMPOUND INTERESTS :

Let P be the principal, R be the interest rate percent per annum, and N be the time period.

1. Simple Interest = (P*N*R)/100
2. Compound Interest = $P(1 + R/100)^N – P$
3. Amount = Principal + Interest

LOGARITHMS :

If $a^m = x$, then $m = \log_a x$.

Properties :

1. $\log_x x = 1$

2. $\log_x 1 = 0$
3. $\log_a(xy) = \log_a x + \log_a y$
4. $\log_a(x/y) = \log_a x - \log_a y$
5. $\log_a x = 1/\log_x a$
6. $\log_a(x^p) = p(\log_a x)$
7. $\log_a x = \log_b x/\log_b a$

Note : Logarithms for base 1 does not exist.

AREA & PERIMETER :

Shape	***Area***	***Perimeter***
Circle	$\prod$ (Radius)^2	2$\prod$(Radius)
Square	(side)^2	4(side)
Rectangle	length*breadth	2(length+breadth)

1. Area of a triangle = 1/2*Base*Height or
2. Area of a triangle = $\sqrt{}$ (s(s-(s-b)(s-c)) where a,b,c are the lengths of the sides and s = (a+b+c)/2
3. Area of a parallelogram = Base * Height
4. Area of a rhombus = 1/2(Product of diagonals)
5. Area of a trapezium = 1/2(Sum of parallel sides)(distance between the parallel sides)
6. Area of a quadrilateral = 1/2(diagonal)(Sum of sides)
7. Area of a regular hexagon = 6($\sqrt{}$3/4)(side)^2
8. Area of a ring = $\prod(R^2-r^2)$ where R and r are the outer and inner radii of the ring.

VOLUME & SURFACE AREA :

Cube :

Let a be the length of each edge. Then,

1. Volume of the cube = a^3 cubic units
2. Surface Area = $6a^2$ square units
3. Diagonal = $\sqrt{3}$ a units

Cuboid :

Let l be the length, b be the breadth and h be the height of a cuboid. Then

1. Volume = lbh cu units
2. Surface Area = 2(lb+bh+lh) sq units
3. Diagonal = $\sqrt{(l^2+b^2+h^2)}$

Cylinder :

Let radius of the base be r and height of the cylinder be h. Then,

1. Volume = $\prod r^2h$ cu units
2. Curved Surface Area = $2\prod rh$ sq units
3. Total Surface Area = $2\prod rh + 2\prod r^2$ sq units

Cone :

Let r be the radius of base, h be the height, and l be the slant height of the cone. Then,

1. $l^2 = h^2 + r^2$
2. Volume = $1/3(\prod r^2h)$ cu units
3. Curved Surface Area = $\prod rl$ sq units
4. Total Surface Area = $\prod rl + \prod r^2$ sq units

Sphere :

Let r be the radius of the sphere. Then,

1. Volume = $(4/3)\prod r^3$ cu units
2. Surface Area = $4\prod r^2$ sq units

Hemi-sphere :

Let r be the radius of the hemi-sphere. Then,

1. Volume = $(2/3)\prod r^3$ cu units
2. Curved Surface Area = $2\prod r^2$ sq units
3. Total Surface Area = $3\prod r^2$ sq units

Prism :

Volume = (Area of base)(Height)

Exercise 1

Solve the following and check with the answers given at the end.

1. It was calculated that 75 men could complete a piece of work in 20 days. When work was scheduled to commence, it was found necessary to send 25 men to another project. How much longer will it take to complete the work?

2. A student divided a number by 2/3 when he required to multiply by 3/2. Calculate the percentage of error in his result.

3. A dishonest shopkeeper professes to sell pulses at the cost price, but he uses a false weight of 950gm. for a kg. His gain is …%.

4. A software engineer has the capability of thinking 100 lines of code in five minutes and can type 100 lines of code in 10 minutes. He takes a break for five minutes after every ten minutes. How many lines of codes will he complete typing after an hour?

5. A man was engaged on a job for 30 days on the condition that he would get a wage of Rs. 10 for the day he works, but he have to pay a fine of Rs. 2 for each day of his absence. If he gets Rs. 216 at the end, he was absent for work for ... days.

6. A contractor agreeing to finish a work in 150 days, employed 75 men each working 8 hours daily. After 90 days, only 2/7 of the work was completed. Increasing the number of men by _____ each working now for 10 hours daily, the work can be completed in time.

7. what is a percent of b divided by b percent of a?

(a) a (b) b (c) 1 (d) 10 (d) 100

8. A man bought a horse and a cart. If he sold the horse at 10 % loss and the cart at 20 % gain, he would not lose anything; but if he sold the horse at 5% loss and the cart at 5% gain, he would lose Rs. 10 in the bargain. The amount paid by him was Rs._____ for the horse and Rs._____ for the cart.

9. A tennis marker is trying to put together a team of four players for a tennis tournament out of seven available. males - a, b and c; females – m, n, o and p. All players are of equal ability and there must be at least two males in the team. For a team of four, all players must be able to play with each other under the following restrictions:

b should not play with m,

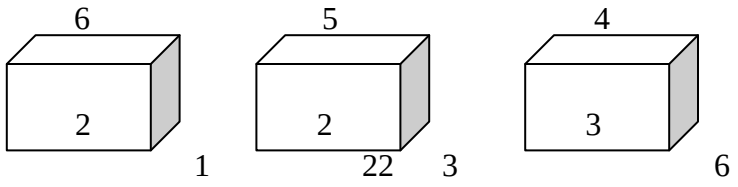
c should not play with p, and

a should not play with o.

Which of the following statements must be false?

1. b and p cannot be selected together
2. c and o cannot be selected together
3. c and n cannot be selected together.

10-12. The following figure depicts three views of a cube. Based on this, answer questions 10-12.



10. The number on the face opposite to the face carrying 1 is _____ .

11. The number on the faces adjacent to the face marked 5 are _____ .

12. Which of the following pairs does not correctly give the numbers on the opposite faces.

(1) 6,5 (2) 4,1 (3) 1,3 (4) 4,2

13. Five farmers have 7, 9, 11, 13 & 14 apple trees, respectively in their orchards. Last year, each of them discovered that every tree in their own orchard bore exactly the same number of apples. Further, if the third farmer gives one apple to the first, and the fifth gives three to each of the second and the fourth, they would all have exactly the same number of apples. What were the yields per tree in the orchards of the third and fourth farmers?

14. Five boys were climbing a hill. J was following H. R was just ahead of G. K was between G & H. They were climbing up in a column. Who was the second?

15-18 John is undecided which of the four novels to buy. He is considering a spy thriller, a Murder mystery, a Gothic romance and a science fiction novel. The books are written by Rothko, Gorky, Burchfield and Hopper, not necessary in that order, and published by Heron, Piegon, Blueja and sparrow, not necessary in that order.

(1) The book by Rothko is published by Sparrow.

(2) The Spy thriller is published by Heron.

(3) The science fiction novel is by Burchfield and is not published by Blueja.

(4)The Gothic romance is by Hopper.

15. Pigeon publishes _____.

16. The novel by Gorky _____.

17. John purchases books by the authors whose names come first and third in alphabetical order. He does not buy the books _____.

18. On the basis of the first paragraph and statement (2), (3) and (4) only, it is possible to deduce that

    1. Rothko wrote the murder mystery or the spy thriller
    2. Sparrow published the murder mystery or the spy thriller
    3. The book by Burchfield is published by Sparrow.

19. If a light flashes every 6 seconds, how many times will it flash in ¾ of an hour?

20. If point P is on line segment AB, then which of the following is always true?

    (1) AP = PB (2) AP > PB (3) PB > AP (4) AB > AP (5) AB > AP + PB

21. All men are vertebrates. Some mammals are vertebrates. Which of the following conclusions drawn from the above statement is correct.

    All men are mammals

    All mammals are men

Some vertebrates are mammals.

None

22. Which of the following statements drawn from the given statements are correct?

Given:

All watches sold in that shop are of high standard. Some of the HMT watches are sold in that shop.

a) All watches of high standard were manufactured by HMT.

b) Some of the HMT watches are of high standard.

c) None of the HMT watches is of high standard.

d) Some of the HMT watches of high standard are sold in that shop.

23-27.

1. Ashland is north of East Liverpool and west of Coshocton.
2. Bowling green is north of Ashland and west of Fredericktown.
3. Dover is south and east of Ashland.
4. East Liverpool is north of Fredericktown and east of Dover.
5. Fredericktown is north of Dover and west of Ashland.
6. Coshocton is south of Fredericktown and west of Dover.

23. Which of the towns mentioned is furthest of the north – west

(Ashland (b) Bowling green (c) Coshocton

(d) East Liverpool (e) Fredericktown

24. Which of the following must be both north and east of Fredericktown?

(Ashland (b) Coshocton (c) East Liverpool

I a only II b only III c only IV a & b V a & c

25. Which of the following towns must be situated both south and west of at least one other town?

A. Ashland only

B. Ashland and Fredericktown

C. Dover and Fredericktown

D. Dover, Coshocton and Fredericktown

E. Coshocton, Dover and East Liverpool.

26. Which of the following statements, if true, would make the information in the numbered statements more specific?

(a) Coshocton is north of Dover.

(b) East Liverpool is north of Dover

(c) Ashland is east of Bowling green.

(d) Coshocton is east of Fredericktown

(e) Bowling green is north of Fredericktown

27. Which of the numbered statements gives information that can be deduced from one or more of the other statements?

(1 (B) 2 (C) 3 (D) 4 (E) 6

28. Eight friends Harsha, Fakis, Balaji, Eswar, Dhinesh, Chandra, Geetha, and Ahmed are sitting in a circle facing the center. Balaji is sitting between Geetha and Dhinesh. Harsha is third to the left of Balaji and second to the right of Ahmed. Chandra is sitting between Ahmed and Geetha and Balaji and Eshwar are not sitting opposite to each other. Who is third to the left of Dhinesh?

29. If every alternative letter starting from B of the English alphabet is written in small letter, rest all are written in capital letters, how the month " September" be written.

(1) SeptEMbEr (2) SEpTeMBEr (3) SeptembeR

(4) SepteMber (5) None of the above.

30. The length of the side of a square is represented by x+2. The length of the side of an equilateral triangle is 2x. If the square and the equilateral triangle have equal perimeter, then the value of x is _____.

31. It takes Mr. Karthik y hours to complete typing a manuscript. After 2 hours, he was called away. What fractional part of the assignment was left incomplete?

32. Which of the following is larger than 3/5?

(1) ½ (2) 39/50 (3) 7/25 (4) 3/10 (5) 59/100

33. The number that does not have a reciprocal is _____.

34. There are 3 persons Sudhir, Arvind, and Gauri. Sudhir lent cars to Arvind and Gauri as many as they had already. After some time Arvind gave as many cars to Sudhir and Gauri as many as they have. After sometime Gauri did the same thing. At the end of this transaction each one of them had 24. Find the cars each originally had.

35. A man bought a horse and a cart. If he sold the horse at 10 % loss and the cart at 20 % gain, he would not lose anything; but if he sold the horse at 5% loss and the cart at 5% gain, he would lose Rs. 10 in the bargain. The amount paid by him was Rs._____ for the horse and Rs._____ for the cart.

Answers:

1. ***Answer:***

30 days.

Explanation:

Before:

One day work	=	1 / 20
One man's one day work	=	1 / (20 * 75)

Now:

No. Of workers	=	50
One day work	=	50 * 1 / (20 * 75)

The total no. of days required to complete the work = (75 * 20) / 50 = 30

2. ***Answer:***

0 %

Explanation:

Since 3x / 2 = x / (2 / 3)

3. ***Answer:***

5.3 %

Explanation:

He sells 950 grams of pulses and gains 50 grams.

If he sells 100 grams of pulses then he will gain (50 / 950) *100 = 5.26

4. ***Answer:***

250 lines of codes

5. ***Answer:***

7 days

Explanation:

The equation portraying the given problem is:

10 * x – 2 * (30 – x) = 216 where x is the number of working days.

Solving this we get x = 23

Number of days he was absent was 7 (30-23) days.

6. ***Answer:***

150 men.

Explanation:

One day's work = 2 / (7 * 90)

One hour's work = 2 / (7 * 90 * 8)

One man's work = 2 / (7 * 90 * 8 * 75)

The remaining work (5/7) has to be completed within 60 days, because the total number of days allotted for the project is 150 days.

So we get the equation

(2 * 10 * x * 60) / (7 * 90 * 8 * 75) = 5/7 where x is the number of men working after the 90^{th} day.

We get x = 225

Since we have 75 men already, it is enough to add only 150 men.

7. ***Answer:***

(c) 1

Explanation:

a percent of b : (a/100) * b

b percent of a : (b/100) * a

a percent of b divided by b percent of a : ((a / 100)*b) / (b/100) * a)) = 1

8. ***Answer:***

Cost price of horse = Rs. 400 & the cost price of cart = 200.

Explanation:-

Let x be the cost price of the horse and y be the cost price of the cart.

In the first sale there is no loss or profit. (i.e.) The loss obtained is equal to the gain.

Therefore (10/100) * x = (20/100) * y

X = 2 * y ----------(1)

In the second sale, he lost Rs. 10. (i.e.) The loss is greater than the profit by Rs. 10.

Therefore (5 / 100) * x = (5 / 100) * y + 10 -------(2)

Substituting (1) in (2) we get

(10 / 100) * y = (5 / 100) * y + 10

(5 / 100) * y = 10

y = 200

From (1) 2 * 200 = **x = 400**

9. ***Answer:***

3.

Explanation:

Since inclusion of any male player will reject a female from the team. Since there should be four member in the team and only three males are available, the girl, n should included in the team always irrespective of others selection.

10. ***Answer:***

5

11. ***Answer:***

1,2,3 & 4

12. ***Answer:***

B

13. ***Answer:***

11 & 9 apples per tree.

Explanation:

Let a, b, c, d & e be the total number of apples bored per year in A, B, C, D & E ‘s orchard. Given that $a + 1 = b + 3 = c – 1 = d + 3 = e – 6$

But the question is to find the number of apples bored per tree in C and D ‘s orchard. If is enough to consider $c – 1 = d + 3$.

Since the number of trees in C's orchard is 11 and that of D's orchard is 13. Let x and y be the number of apples bored per tree in C & d 's orchard respectively.

Therefore 11 x – 1 = 13 y + 3

By trial and error method, we get the value for x and y as 11 and 9

14. ***Answer:***

G.

Explanation:

The order in which they are climbing is R – G – K – H – J

15 – 18

Answer:

Novel Name	Author	Publisher
Spy thriller	Rathko	Heron
Murder mystery	Gorky	Piegon
Gothic romance	Burchfield	Blueja
Science fiction	Hopper	Sparrow

Explanation:

Given

Novel Name	Author	Publisher
Spy thriller	Rathko	Heron
Murder mystery	Gorky	Piegon

Gothic romance	Burchfield	Blueja
Science fiction	Hopper	Sparrow

Since Blueja doesn't publish the novel by Burchfield and Heron publishes the novel spy thriller, Piegon publishes the novel by Burchfield.

Since Hopper writes Gothic romance and Heron publishes the novel spy thriller, Blueja publishes the novel by Hopper.

Since Heron publishes the novel spy thriller and Heron publishes the novel by Gorky, Gorky writes Spy thriller and Rathko writes Murder mystery.

19. ***Answer:***

451 times.

Explanation:

There are 60 minutes in an hour.

In ¾ of an hour there are (60 * ¾) minutes = 45 minutes.

In ¾ of an hour there are (60 * 45) seconds = 2700 seconds.

Light flashed for every 6 seconds.

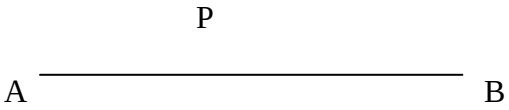
In 2700 seconds 2700/6 = 450 times.

The count start after the first flash, the light will flashes 451 times in ¾ of an hour.

20. ***Answer:***

(4)

Explanation:

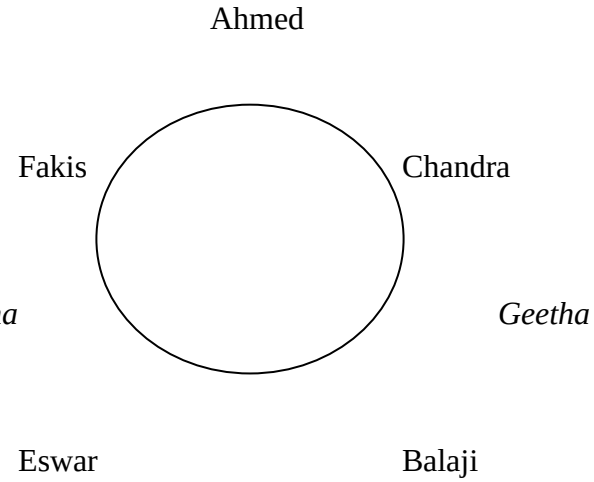


Since p is a point on the line segment AB, AB > AP

21. ***Answer:*** (c)

22. ***Answer:*** (b) & (d).

23 - 27.***Answer:***



28. ***Answer:*** Fakis

Explanation:

29. ***Answer:***

(5).

Explanation:

Since every alternative letter starting from B of the English alphabet is written in small letter, the letters written in small letter are b, d, f...

In the first two answers the letter E is written in both small & capital letters, so they are not the correct answers. But in third and fourth answers the letter is written in small letter instead capital letter, so they are not the answers**.**

30. ***Answer:***

$x = 4$

Explanation:

Since the side of the square is $x + 2$, its perimeter = $4 (x + 2) = 4x + 8$

Since the side of the equilateral triangle is $2x$, its perimeter = $3 * 2x = 6x$

Also, the perimeters of both are equal.

(i.e.) $4x + 8 = 6x$

(i.e.) $2x = 8 \rightarrow x = 4$.

31. ***Answer:***

$(y – 2) / y$.

Explanation:

To type a manuscript karthik took y hours.

Therefore his speed in typing = 1/y.

He was called away after 2 hours of typing.

Therefore the work completed = 1/y * 2.

Therefore the remaining work to be completed = 1 – 2/y.

(i.e.) work to be completed = (y-2)/y

32. ***Answer:***

(2)

33. ***Answer:***

1

Explanation:

One is the only number exists without reciprocal because the reciprocal of one is one itself.

34. ***Answer:***

Sudhir had 39 cars, Arvind had 21 cars and Gauri had 12 cars.

Explanation:

	Sudhir	Arvind	Gauri
Finally	24	24	24
Before Gauri's transaction	12	12	48

Before Arvind's transaction	6	42	24
Before Sudhir' s transaction	39	21	12

35. ***Answer:***

Cost price of horse: Rs. 400 &

Cost price of cart: Rs. 200

Explanation:

Let x be the cost of horse & y be the cost of the cart.

10 % of loss in selling horse = 20 % of gain in selling the cart

Therefore (10 / 100) * x = (20 * 100) * y

➔ x = 2y ----------(1)

5 % of loss in selling the horse is 10 more than the 5 % gain in selling the cart.

Therefore (5 / 100) * x - 10 = (5 / 100) * y

➔ 5x - 1000 = 5y

Substituting (1)

10y - 1000 = 5y

5y = 1000

y = 200

x = 400 from (1)

Exercise 2.1

For the following, find the next term in the series

1. 6, 24, 60,120, 210

336 b) 366 c) 330 d) 660

Answer: 336

Explanation:

The series is 1.2.3, 2.3.4, 3.4.5, 4.5.6, 5.6.7, ... ('.' means product)

2. 1, 5, 13, 25

Answer: 41

Explanation:

The series is of the form 0^2+1^2, 1^2+2^2,...

3. 0, 5, 8, 17

Answer: 24

Explanation:

1^2-1, 2^2+1, 3^2-1, 4^2+1, 5^2-1

4. 1, 8, 9, 64, 25 (Hint : Every successive terms are related)

Answer: 216

Explanation:

1^2, 2^3, 3^2, 4^3, 5^2, 6^3

5. 8,24,12,36,18,54

Answer: 27

6 71,76,69,74,67,72

Answer: 67

7. 5,9,16,29,54

Answer: 103

Explanation:

5*2-1=9; 9*2-2=16; 16*2-3=29; 29*2-4=54; 54*2-5=103

8. 1,2,4,10,16,40,64 (Successive terms are related)

Answer: 200

Explanation:

The series is powers of 2 (2^0,2^1,..).

All digits are less than 8. Every second number is in octal number system.

128 should follow 64. 128 base 10 = 200 base 8.

Exercise 2.2

Find the odd man out.

1. 3,5,7,12,13,17,19

Answer: 12

Explanation:

All but 12 are odd numbers

2. 2,5,10,17,26,37,50,64

Answer: 64

Explanation:

2+3=5; 5+5=10; 10+7=17; 17+9=26; 26+11=37; 37+13=50; 50+15=65;

3. 105,85,60,30,0,-45,-90

Answer: 0

Explanation:

105-20=85; 85-25=60; 60-30=30; 30-35=-5; -5-40=-45; -45-45=-90;

Exercise 3

Solve the following.

1. What is the number of '0' at the end of the product of the numbers from 1 to 100?

Answer: 127

2. A fast typist can type some matter in 2 hours and a slow typist can type the same in 3 hours. If both type combinely, in how much time will they finish?

Answer: 1 hr 12 min

Explanation:

The fast typist's work done in 1 hr = 1/2

The slow typist's work done in 1 hr = 1/3

If they work combinely, work done in 1 hr = 1/2+1/3 = 5/6

So, the work will be completed in 6/5 hours. i.e., 1+1/5 hours = 1hr 12 min

3. Gavaskar's average in his first 50 innings was 50. After the 51st innings, his average was 51. How many runs did he score in his 51st innings. (supposing that he lost his wicket in his 51st innings)

Answer: 101

Explanation:

Total score after 50 innings = 50*50 = 2500

Total score after 51 innings = 51*51 = 2601

So, runs made in the 51st innings = 2601-2500 = 101

If he had not lost his wicket in his 51st innings, he would have scored an unbeaten 50 in his 51st innings.

4. Out of 80 coins, one is counterfeit. What is the minimum number of weighings needed to find out the counterfeit coin?

Answer: 4

5. What can you conclude from the statement : All green are blue, all blue are red. ?

i. some blue are green

ii. some red are green

iii. some green are not red

iv. all red are blue

1. i or ii but not both
2. i & ii only
3. iii or iv but not both
4. iii & iv

Answer: (2)

6. A rectangular plate with length 8 inches, breadth 11 inches and thickness 2 inches is available. What is the length of the circular rod with diameter 8 inches and equal to the volume of the rectangular plate?

Answer: 3.5 inches

Explanation :

Volume of the circular rod (cylinder) = Volume of the rectangular plate

(22/7)*4*4*h = 8*11*2

h = 7/2 = 3.5

7. What is the sum of all numbers between 100 and 1000 which are divisible by 14 ?

Answer: 35392

Explanation:

The number closest to 100 which is greater than 100 and divisible by 14 is 112, which is the first term of the series which has to be summed.

The number closest to 1000 which is less than 1000 and divisible by 14 is 994, which is the last term of the series.

112 + 126 + ... + 994 = 14(8+9+ ... + 71) = 35392

8. If s(denotes square root of a, find the value of s(12+s(12+s(12+ ..... upto infinity.

Answer: 4

Explanation :

Let x = s(12+s(12+s(12+.....

We can write x = s(12+x). i.e., x^2 = 12 + x. Solving this quadratic equation, we get x = -3 or x=4. Sum cannot be -ve and hence sum = 4.

9. A cylindrical container has a radius of eight inches with a height of three inches. Compute how many inches should be added to either the radius or height to give the same increase in volume?

Answer: 16/3 inches

Explanation:

Let x be the amount of increase. The volume will increase by the same amount if the radius increased or the height is increased. So, the effect on increasing height is equal to the effect on increasing the radius.

i.e., (22/7)*8*8*(3+x) = (22/7)*(8+x)*(8+x)*3

Solving the quadratic equation we get the x = 0 or 16/3. The possible increase would be by 16/3 inches.

10. With just six weights and a balance scale, you can weigh any unit number of kgs from 1 to 364. What could be the six weights?

Answer: 1, 3, 9, 27, 81, 243 (All powers of 3)

11. Diophantus passed one sixth of his life in childhood, one twelfth in youth, and one seventh more as a bachelor; five years after his marriage a son was born who died four years before his father at half his final age. How old is Diophantus?

Answer: 84 years

Explanation:

$x/6 + x/12 + x/7 + 5 + x/2 + 4 = x$

12. If time at this moment is 9 P.M., what will be the time 23999999992 hours later?

Answer: 1 P.M.

Explanation:

24 billion hours later, it would be 9 P.M. and 8 hours before that it would be 1 P.M.

13. How big will an angle of one and a half degree look through a glass that magnifies things three times?

Answer: 1 1/2 degrees

Explanation:

The magnifying glass cannot increase the magnitude of an angle.

14. Divide 45 into four parts such that when 2 is added to the first part, 2 is subtracted from the second part, 2 is multiplied by the third part and the fourth part is divided by two, all result in the same number.

Answer: 8, 12, 5, 20

Explanation:

a + b + c + d =45; a+2 = b-2 = 2c = d/2; a=b-4; c = (b-2)/2; d = 2(b-2); b-4 + b + (b-2)/2 + 2(b-2) = 45;

15. I drove 60 km at 30 kmph and then an additional 60 km at 50 kmph. Compute my average speed over my 120 km.

Answer: 37 1/2

Explanation:

Time reqd for the first 60 km = 120 min.; Time reqd for the second 60 km = 72 min.; Total time reqd = 192 min

Avg speed = (60*120)/192 = 37 1/2

Questions 16 and 17 are based on the following :

Five executives of European Corporation hold a Conference in Rome Mr. A converses in Spanish & Italian

Mr. B, a spaniard, knows English also

Mr. C knows English and belongs to Italy

Mr. D converses in French and Spanish

Mr. E , a native of Italy knows French

16. Which of the following can act as interpreter if Mr. C & Mr. D wish to converse

only Mr. A b) Only Mr. B c) Mr. A & Mr. B d) Any of the other three

Answer: d) Any of the other three.

Explanation:

From the data given, we can infer the following.

A knows Spanish, Italian

B knows Spanish, English

C knows Italian, English

D knows Spanish, French

E knows Italian, French

To act as an interpreter between C and D, a person has to know one of the combinations Italian & Spanish, Italian & French, English & Spanish, English & French. A, B, and E know atleast one of the combinations.

17. If a 6th executive is brought in, to be understood by maximum number of original five he should be fluent in

English & French b)Italian & Spanish c)English & French d) French & Italian

Answer: b) Italian & Spanish

Explanation:

Number of executives who know

i) English is 2

ii) Spanish is 3

iii) Italian is 3

iv) French is 2

Italian & Spanish are spoken by the maximum no of executives. So, if the 6th executive is fluent in Italian & Spanish, he can communicate with all the original five because everybody knows either Spanish or Italian.

18. What is the sum of the first 25 natural odd numbers?

Answer: 625

Explanation:

The sum of the first n natural odd nos is square(n).

1+3 = 4 = square(2) 1+3+5 = 9 = square(3)

19. The sum of any seven consecutive numbers is divisible by

a) 2 b) 7 c) 3 d) 11

Answer:

(b) 7

Explanation:

Let x be any number. The next six consecutive numbers are x+1, x+2, x+3, x+4, x+5 and x+6. The sum of these seven numbers are 7x + 21. This is equal to 7(x+3). This number will always divisible by 7. Hence the result.

Exercise 3

Try the following.

1. *There are seventy clerks working in a company, of which 30 are females. Also, 30 clerks are married; 24 clerks are above 25 years of age; 19 married clerks are above 25 years, of which 7 are males; 12 males are above 25 years of age; and 15 males are married. How many bachelor girls are there and how many of these are above 25?*

2. *A man sailed off from the North Pole. After covering 2,000 miles in one direction he turned West, sailed 2,000 miles, turned North and sailed ahead another 2,000 miles till he met his friend. How far was he from the North Pole and in what direction?*

3. *Here is a series of comments on the ages of three persons J, R, S by themselves.*

S : The difference between R's age and mine is three years.

J : R is the youngest.

R : Either I am 24 years old or J 25 or S 26.

J : All are above 24 years of age.

S : I am the eldest if and only if R is not the youngest.

R : S is elder to me.

J : I am the eldest.

R : S is not 27 years old.

S : The sum of my age and J's is two more than twice R's age.

One of the three had been telling a lie throughout whereas others had spoken the truth. Determine the ages of S,J,R.

4. *In a group of five people, what is the probability of finding two persons with the same month of birth?*

5. *A father and his son go out for a 'walk-and-run' every morning around a track formed by an equilateral triangle. The father's walking speed is 2 mph and his running speed is 5 mph. The son's walking and running speeds are twice that of his father. Both start together from one apex of the triangle, the son going clockwise and the father anti-clockwise. Initially the father runs and the son walks for a certain period of time. Thereafter, as soon as the father starts walking, the son starts running. Both complete the course in 45 minutes. For how long does the father run? Where do the two cross each other?*

6. *The Director of Medical Services was on his annual visit to the ENT Hospital. While going through the out patients' records he came across the following data for a particular day : " Ear consultations 45; Nose 50; Throat 70; Ear and Nose 30; Nose and Throat 20; Ear and Throat 30; Ear, Nose and Throat 10; Total patients 100." Then he came to the conclusion that the records were bogus. Was he right?*

7. *Amongst Ram, Sham and Gobind are a doctor, a lawyer and a police officer. They are married to Radha, Gita and Sita (not in order). Each of the wives have a profession. Gobind's wife is an artist. Ram is not married to Gita. The lawyer's wife is a teacher. Radha is married to the police officer. Sita is an expert cook. Who's who?*

8. *What should come next?*

1, 2, 4, 10, 16, 40, 64,

Questions 9-12 are based on the following :

Three adults – Roberto, Sarah and Vicky – will be traveling in a van with five children – Freddy, Hillary, Jonathan, Lupe, and Marta. The van has a driver's seat and one passenger seat in the front, and two benches behind the front seats, one beach behind the other. Each bench has room for exactly three people. Everyone must sit in a seat or on a bench, and seating is subject to the following restrictions: An adult must sit on each bench.

Either Roberto or Sarah must sit in the driver's seat.

Jonathan must sit immediately beside Marta.

9. *Of the following, who can sit in the front passenger seat ?*

(Jonathan (b) Lupe (c) Roberto (d) Sarah (e) Vicky

10. Which of the following groups of three can sit together on a bench?

(Freddy, Jonathan and Marta (b) Freddy, Jonathan and Vicky

(c) Freddy, Sarah and Vicky (d) Hillary, Lupe and Sarah

(e) Lupe, Marta and Roberto

11. If Freddy sits immediately beside Vicky, which of the following cannot be true ?

a. *Jonathan sits immediately beside Sarah*
b. *Lupe sits immediately beside Vicky*
c. *Hillary sits in the front passenger seat*

d. *Freddy sits on the same bench as Hillary*

e. *Hillary sits on the same bench as Roberto*

12. *If Sarah sits on a bench that is behind where Jonathan is sitting, which of the following must be true ?*

a. *Hillary sits in a seat or on a bench that is in front of where Marta is sitting*

b. *Lupe sits in a seat or on a bench that is in front of where Freddy is sitting*

c. *Freddy sits on the same bench as Hillary*

d. *Lupe sits on the same bench as Sarah*

e. *Marta sits on the same bench as Vicky*

13. *Make six squares of the same size using twelve match-sticks. (Hint : You will need an adhesive to arrange the required figure)*

14. *A farmer has two rectangular fields. The larger field has twice the length and 4 times the width of the smaller field. If the smaller field has area K, then the are of the larger field is greater than the area of the smaller field by what amount?*

(6K (b) 8K (c) 12K (d) 7K

15. *Nine equal circles are enclosed in a square whose area is 36sq units. Find the area of each circle.*

16. *There are 9 cards. Arrange them in a 3*3 matrix. Cards are of 4 colors. They are red, yellow, blue, green. Conditions for arrangement: one red card must be in first row or second row. 2 green cards should be in 3rd column. Yellow cards must be in the 3 corners only. Two blue cards must be in the 2nd row. At least one green card in each row.*

17. *Is z less than w? z and w are real numbers.*

(I) $z^2 = 25$

(II) w = 9

To answer the question,

Either I or II is sufficient

b) Both I and II are sufficient but neither of them is alone sufficient

c) I & II are sufficient

d) Both are not sufficient

18. *A speaks truth 70% of the time; B speaks truth 80% of the time. What is the probability that both are contradicting each other?*

19. *In a family 7 children don't eat spinach, 6 don't eat carrot, 5 don't eat beans, 4 don't eat spinach & carrots, 3 don't eat carrot & beans, 2 don't eat beans & spinach. One doesn't eat all 3. Find the no. of children.*

20. *Anna, Bena, Catherina and Diana are at their monthly business meeting. Their occupations are author, biologist, chemist and doctor, but not necessarily in that order.*

Diana just told the neighbour, who is a biologist that Catherina was on her way with doughnuts. Anna is sitting across from the doctor and next to the chemist. The doctor was thinking that Bena was a good name for parent's to choose, but didn't say anything. What is each person's occupation?

Exercise 4.

1. *Krishna, Hari and Prakash went for a race. Krishna gives Hari a start of 20% of his distance, similarly gives Krishna a start of 20% of his distance. Prakash reached the designation within 80 sec. whose speed is half that of Krishna. But Hari is twice as fast as Prakash. By what time after Hari, Krishan reaches the designation.*

2. *In a km race Siddharth beats Thanigai by 20% of the race distance. If Thanigai takes 20 sec more than Siddharth then by what is the difference in their speeds?*

3. *In a 200m race between Karthik & Arasu, Arasu can give Karthik a start of 20m in the first 100m. When Karthik crosses the 80m mark both of them changes their speeds such that Karthik can give Arasu a start of 20m in the next 100m. They complete the race with that speed. Who is the winner of the race and by how many seconds?*

4. *In a race between Vimal, Shree Hari and Varadha, Shree Hari can give Vimal 20% and Varadha 10% of their distance. then Varadha can give Vimal a start of _____% of his distance.*

5. *In a cricket match Madan chases the ball, which is away from him by 20% of the distance between him and boundary. If the speed of Madan is 25% more than that of the ball then they will reach the boundary at the same time. If the speed of the ball is 20m/s then what is the distance between the ball and boundary?*

6. *Ganesh & Thiagarajan and Suresh & Sakthivel were the two teams for a 2*100 m relay. In a hundred-meter race, Ganesh beats Suresh by 20m and Sakthivel beats Thiagarajan by 20m. The ratio of speeds of Suresh & Thiagarajan is 2:3. Suresh and Ganesh started the race. The winning team is _____. They win the race beating the losing team by _____.*

7. *The two contestants of a car rally were Arvind and Abilash. Arvind starts 5sec later and beats Abilash by 5sec. The quickest person goes with a speed twice that of the other. If Abilash was 100m behind when Arvind finishes the race then what is the distance that Abilash has to cover to finish the game when Arvind begins the race?*

8. *In a circular athletic ground Carl & Gauri started their race standing half a way from each other. The minimum distance between them is 28m. They have to complete the race at the same winning point, the Carl's starting point. When the race was over Carl completed three full rounds & Gauri two. If both of them reached the goal at the same time what is the ratio between the speed of Carl & Gauri*

.

9. *In a 1500-meter race, in a 100m circular track, between Guru and Selvam due to wrong judgement Guru beats Selva by 10sec. The mistake made by the judges was that they counted*

one round extra to Guru. The sum of time taken by them is 90sec. If the mistake was not made, then who is going to win the race & by how many seconds?

10. *In a game of 80 points, Sudhir can give Santhosh 5 points and Shree Ram 15 points. Then how many points Santhosh can give Shree Ram in a game of 60 points?*