C++

Contents

Basic

1) A relational operator
a) assigns one operand to another
b) **Compares two operands**
c) Logically combincs two operands
d) assigns value to a operand

2) The Library function exit() causes an exit from
a) the loop in which it occurs
b) the block in which it occurs
c) the function in which it occurs
d) **The program in which it occurs**

3) The && and || operators
a) compare two numeric values
b) combine two numeric values
c) **compare two Boolean values**
d) combine two Boolean values

4) The goto statement causes control to go to
a) an operator b) **a label** c) a variable d) a function

5) The break statement cause an exit
a) only from the innermost loop
b) only from the innermost switch
c) from all loops and switches
d) **from the innermost loop or switch**

6) A structure brings together a group of
a) items of same data type
b) **related data items**
c) integers with user-defined names
d) constant values

7) A functions argument is
a) **a variable in the function that receives a value from the calling program**
b) a reference value returned by the function
c) a value sent to function by the calling program
d) a value returned by the function to the calling program

8) When an argument is passed by reference
a) a variable is created in the function to hold the arguments value
b) the function cannot access the arguments value
c) a temporary variable is created in the calling program to hold the arguments value

d) **the function accesses the arguments original value in the calling program**

9) Which will be the output of the following program?

```
int main()
{
 Int arr[] = {10,11,12,13,14};
int *p =(arr +1);
cout << *arr +5;
return 0;
}
```

a) **15** b) 16 c) 14 d) Compile error

10) When an argument is passed by reference
a) A variable is created in the function to hold the arguments value
b) The function cannot access the argument value
c) A temporary variable is created in the calling program to hold the argument value
d) **The function accesses the argument original value in the calling program**

11) Which among the following is an exit controlled loop
a) if b) **do-while** c) while d) for

12) What will be the storage class of variable in the following code

```
int main()
{
int i=1;
cout<<i;
 return 0;
 }
```

a) **Automatic storage class** b) External storage class
c) Static storage class d) Register storage class

Enhancements

1) What is the output?

```
const int a=124;
void main()
{
        const int* sample();
 int * const p=sample();
 cout<<*p;
}
const int* sample()
 {      return (&a);
}
```

a) Warning b) **compilation error** c) output "124" d) garbage value

2) What is the output?

```
#include<iostream.h>
void accept(int x,int y)
{
        cout<<"in value method\n";
```

```
}
void accept(int &p,int &q)
{
        cout<<"in referece method\n";
}
void main()
{
  Int a=20,b=30;
        accept(a,b);
}
```

a) output "in reference method **b) compilation error**
c) output "in value method in reference method" d) output "in value method"

3) What is the output?

```
void fun(int ptr2)
{
        ptr2=30;
}
void main()
{
        int num=10;
        fun(num);
        cout<<num<<endl;
        getch();
}
```

a) 10 b) garbage value c) it will not compile d) 30

4) What is the output?

```
void main()
{
    int* getAr();
     int *ptr;
     ptr=getAr();
     cout<<ptr[2]<<endl;
     getch();
}
int* getAr()
{
     int arr[4]={10,20,30,40};
     return arr;
}
```

a) 20 b) 30 c) it will not compile d) **warning**

5) In case of command line arguments main accepts following two arguments.
a) int argc,char *argv b) char argv,int argc
c) int argc,char *argv[] d) char *argv,int *argc

6) It is legal to return local variables from a function, through reference.
a) True **b) False**

7) In C++ one can define a function within another function
a) True b) False

8) In C++ an identifier can begin with a $ sign
a) **True** b) False

9) What is the output?
```
#include<iostream.h>
int a = 1;
void main()
{
   int a = 100;
    {
        int b = 200;
   {
int a = 300;
            cout<<a<<",";
        }
   cout<<a<<",";
cout<<a<<",";
        }
}
```
a) 100 300 100 b) Error **c) 300 100 100** d) 300 100 garbage

10) What will happen to following code?
```
 struct emp
{
        char name[20];
};
void main()
{
 emp e1={"abc"};       emp
 e2=e1;
        cout<<e2.name<<en
 dl;    getch();
}
```
a) warning b) compiler error "can not initialize e2 with e1"
c) output "abc" d) garbage

11) Which statement will print the value of num ?
```
struct mystruct
{
 int *k;
};
void main()
{
 int num=200;
mystruct *ptr=new mystruct;
 ptr->k=&num;  //
 here
 getch();
}
```
a) *(*ptr).k or *ptr->k b) *ptr.k c) ptr->k d) ptr->*k

12) The _____ operator allows conversion between nonstandard types.

a) reinterpret_cast b) const_cast c) static_cast d) None of the above

13) *p++ ;
a) increments value **b) increments address** c) Error d) None

14) The statements
int a=5;
cout<<"First"<<(a<<2)<<"Second"; Output will be
a) First52Second **b) First20Second** c) Second25First d) An error message.

15) The following program segment int a =10; int const &b=a; a=11
printf("%d%d",a,b);
a) Results in compile time error b) Results in run time error
c) 1 1 1 1 d) None of the above.

16) What will be the output ?

```
#include<iostream.h>
void main()
{
 int a,*pa,&ra;
      pa=&a;  ra=a;
 cout<<"a="<<a<<"pa="<<pa<<"ra"<<ra;
    }
```

a) compile time error b) runtime error c) will display correct output d) none of the above

17) What is the output ?

```
#include<iostream.h>
void main()
{
        int arr[2][3][2]={{{2,4},{4,8},{3,4},},{{2,2},{2,3},{3,4},}};
        cout<<**(*arr+1)+2+7;
}
```

a) 7 **b) 13** c) 16 d) Error

18)What is the output? void main()

```
{
        int arr[2][3][2]={{{2,4},{4,8},{3,4},},{{2,2},{2,3},{3,4},}};
        cout<<***(arr+1)+5+4;
}
```

a) 12 b) 25 **c) 11** d) None of these
Explanation:
***(arr+1)+5+4
Solve *(arr+1) , this is equivalent to arr[1] i.e. base address of second dd array.
Add one more *, u will get address of first one d array represented by second dd array.
Add one more *, u will get an element of first one d array represented by second dd array i.e. 2
Now
2+5+4
i.e. 11.

19) int f()
{

```
    int
i=12;
    int
&r=i;
    r+=r/4;
    int
*p=&r;
    *p+=r;
        return i;
}
```

Referring to the sample code above , what is the return value of the function "f()" ?

a) 15 **b) 30** c) 24 d) 12

20) Inline functions are replaced at function call
at the time of

a) preprocessing b) runtime **c) compiletime** d) unpredictable

21) what is the output ?

```
#include<stdio.h>
void main()
{
    int x=4;
    printf("%d",printf("%d%d",x,x) );
}
```

a) Garbage **b) 4,4,2** c) 2,2,4 d) compile time error

22) consider following code

```
#include<iostream.h>
    void main()
    {
    in
    t
    i,j
    ;
     for(i=0;i<2;i++)
     {
        for(j=0;j<3;j++)
        {
            if(i==j)
            {
                continue;
            }
            cout<<"i="<<i<<"j="<<j<<endl;
        }
     }
    }
```

For which values of i and j the above code will not give any output ?

a) i=1 j=0 **b) i=0 j=0** c) i=0 j=2 d) i=0 j=1

23) Consider the following code.

```
#include<iostream.h>
```

```
#include<string.h>
#include<stdlib.h>
void ReadInput(int DataType,void *address)
{
        char buffer[30];
        cin.getline(buffer,sizeof(buffer));
        switch(DataType)
        {
        case 1:
                *(int*)address=atoi(buffer);
        break;
    case 2:
                *(float*)address=atof(buffer);
        break;
    case 3:
                strcpy((char*)address,buffer);
                break;
        }
}
void main()
{
        float x;
    cout<<"\nEnter number\n";
    ReadInput(2,&x);
        cout<<"\nsquare=" <<x*x;
}
```

What would be output if input provided is 12.5

a) 156.25 b) compile time error. Cannot convert from float to int

c) 144 d) none of the above.

24) what is the output ?

```
#include<iostream.h>
void main()
{
    int a=20;
    int &n=a;
    n=a++;
    a=n++;
        cout<<a<<"\t"<<n<<endl;
}
```

a) 20 20 b) 20 21 c) 21 22 **d) 22 22**

25) what is the output ?

```
#include<iostream.h>
void main()
{
    int arr[]={10,20,30,40,50};
    int x,*ptr1=arr,*ptr2=&arr[3];
    x=ptr2-ptr1;
    cout<<x<<endl;
}
```

a) 6 **b) 3** c) compile time error d)runtime error

26) what is the output ? #include<iostream.h>
void main()
{
int a=20 ,b=100; int &n=a; n=a++;
n=&b;
cout<<a<<"\t"<<n<<endl;
}
a) 20 21 b) 21 20 c) 21 22 **d) Error**

27) in case of command line arguments main accepts following two arguments.
a) int argc,char *argv b) char argv,int argc
c) int argc,char *argv[] d) char *argv,int *argc

28) using which macro, we can display the argument from variable number of argument function ?.
a) va_arg b) va_list c) va_show d) va_start

29) What is the output ? void fun(int *ptr2)
{
*ptr2=30;
}
void main()
{
int num=10; int *ptr1=# fun(ptr1); cout<<num<<endl; getch();
}
a) 10 b) garbage value c) it will not compile **d) 30**

30) what is the output ?
void main()
{
int* getAr();
int *ptr;
ptr=getAr();
cout<<ptr[2]<<en
dl; getch();
}
int* getAr()
{
int arr[4]={10,20,30,40};
return arr;
}
a) 20 b) 30 c) it will not compile **d) warning**

31) What will happen to following code ?
struct emp
{
char name[20];
};
void main()

```
{
 emp e1={"abc"};
emp e2;
e2.name=e1.name;
cout<<e2.name<<e
ndl;
 getch();
}
```

a) warning b) **compiler error** c) output "abc" d) none of the above.

32) which statement will print the value of num ? struct mystruct

```
{
        int *k;
};
        void main()
{
        int num=200;
        mystruct *ptr=new mystruct;

ptr-
>k=&num;
// here
getch();
}
```

a) *(*ptr).k or *ptr->k b) *ptr.k c) ptr->k d) ptr->*k

33) What is the output ?

```
const int a=124;
    void main()
    {
        const int* sample();    int *p;
        p=sample();
        cout<<*p;
    }
    const int* sample()
        {       return (&a);
}
```

a) warning **b) compilation error** c) output "124" d) garbage value

34) For the following allocation which would be the proper deallocation?

```
int *p = new int[5]
```

a) Free(p) b) Delete p **c) Delete [] p** d) None of the above

35) References are allocated memory

a) False b) True

36) If ptr is a pointer to array of objects, then delete ptr and delete [] ptr both are same

a) False b) True

37) Which one of the following is demonstrated by the sample code above?

a) A default function parameter b) A virtual member function

c) A template function d) A member function definition

38) The statements
int a=5;
cout<<"First"<<(a<<2)<<"Second";
Output will be
a) First52Second b) Second25First
c) First20Second d) An error message.

39) The following program segment int a =10; int const &b=a; a=11
printf("%d%d",a,b);
a) Results in compile time error b) Results in run time error
c) 1 1 1 1 d) None of the above.

40) int f()
{
int
i=12;
int
&r=i;
r+=r/4
; int
*p=&r
;
*p+=r;
return i;
}
Referring to the sample code above , what is the return value of the function "f()" ?
a) 15 **b) 30** c) 24 d) 12

41) What is the output ? #include<stdio.h> void main()
{
int x=4;
printf("%d",printf("%d%d",x,x));
}
a) Garbage **b) 4,4,2** c) 2,2,4 d) compile time error

42) What is the output ? #include<iostream.h>
void main()
{
int a=20; int &n=a; n=a++; a=n++;
cout<<a<<"\t"<<n<<endl;
}
a) 20 20 b) 20 21 c) 21 22 **d) 22 22**

43) What is the output ? #include<iostream.h>
void main()
{
int arr[]={10,20,30,40,50};
int x,*ptr1=arr,*ptr2=&arr[3];
x=ptr2-ptr1;

```
    cout<<x<<endl;
}
```

a) 6 **b) 3** c) compile time error d) runtime error

44) Identify following
a) const int * ptr; _____
int const * str; _____

45) We can not make constant pointer pointing to constant int variable.
a) **True** b) False

46) Array of reference can not be created.
a) **True** b) False

47) Using which macro, we can initialize the list of data in case of variable number of argument function?
a) va_arg b) va_list c) va_show **d) va_start**

48) In C++ function call can be on left side.
a) **True** b) False

49) We can make pointer to constant pointing to non-constant int variable.
a) **True** b) False

50) cin and cout are present in
a) stdio.h **b) iostream** c) conio.h

51) Name mangling always happens in C++.
a) false **b) true**

52) In case of function overloading
a) arguments must be different , return type may or may not be different
b) return type must be different,arguments may or may not be different
c) both return type and arguments must be same
d) both return type and arguments must be different

53) What will happen to the following code while compiling ?

```
int& retVal()
{
int cnt=20;
 return cnt;
}
```

a) No Error b) Error **c) Warning**

54)

```
#include<iostream.h>
void main()
{
char * const t="hello";
t="world";
}
```

a) Runtime Error b) **Compilation Error** c) Neither Compilation or Runtime Error

55)
```
#include<iostream.h>
int& disp()
{
    int num=10;
    return num;
}
void main()
{
  disp()=30;
}
```
a) Compilation Error b) No Error,No Warning c) **Warning**

56)
```
#include<iostream.h> void main()
{
    int i=5;
    int
&j=i;
    int
p=10;
    j=p;
        p=20;
            cout<<endl<<i<<endl<<j;
}
```
a) 20,20 b) 10,5 c) 5,10 d) **10,10**

57)
```
#include<iostream.h> void main()
{
            char *p="Hello";
        char *q=p;
            q="Good Bye";
            cout<<p<<"\t"<<q;
}
```
a) **Hello Good Bye** b) Good Bye Good Bye c) Error: Lvalue Reqd.

58)
```
#include<iostream.h
> const int a=124;
void main()
    {
        const int* sample();
        int *p;
        p=sample();
    }
    const int* sample()
{       return (&a);
    }
```
a) Warning b) Neithe Warning nor Error c) **Compilation Error**

59)
```
#include<iostream>
void main()
{
```

```
        char t[]="String functions are simple";
        int len=strlen(t);
        cout<<len;
}
```

a) **Compilation Error** b) Warning c) successful output

```
60)     #include<iostream>
        void main()
        {
                int a=30;
                f();
        }
    void f()
    {
                int b=30;
}
```

a) Successful output b) Warning c) **Compilation Error**

61) What will happen to the following code ?

```
#include<iostream.h
> void main()
{ for(int
i=0;i<5;i++)
{       int a=0;
a++;
}
cout<<endl<<a;
}
```

a) **compilation error** b) it will print garbage value c) it will print 1 d) it will print 5

62) what will happen to the following code ?

```
#include<iostream.h>
        void main()
        {
        for(int i=0;i<5;i++)
        {
        cout<<endl<<i;
        }
    for(int i=5;  ;i++)
        {
        cout<<endl<<i;
        }
        }
```

a) it will print 0 to 9
b) infinite loop because there is no condition in second for loop
c) **compilation error**

63) C++ compiler internally changes names of all functions at the declaration, definition and call. This process is known as _____ **or** _____**.**

64) True or False. Default arguments can be given in the beginning or in between also.

a) True b) **False**

65) Function overloading and operator overloading comes under
a) Run time polymorphism **b) Compile time polymorphism**
c) Both a and b are correct d) None of the above

66) What will be the output of the following code?

```
#include<iostream.h>
#define MAXROW 3
   #define MAXCOL 4
   void main()
{
int (*p) [MAXCOL]; p=new
   int[MAXROW][MAXCOL];
cout<<endl<<sizeof(p)<<endl<<sizeof(*p);
}
```

a) 2(under Dos) or 4(under Linux or windows) 8(under Dos) or 16(under Linux or windows)
b) 4(under Dos) or 8(under Linux or windows) 8(under Dos) or 16(under Linux or windows)
c) compilation error
d) runtime error

67) What is the output of the program?

```
#include <iostream.h>
void main ()
{
for(int j = 1, sum = 0; j < 5; j++)
   sum += j;
sum = j;
cout << sum;
}
```

a) 6 **b) 5** c) Compilation error. Undefined variable sum and j d) 10

68) Which of the following is false about struct and class in C++?
a) he members of a struct are public by default, while in class, they are private by default
b) Struct and class are otherwise functionally equivalent
c) A class supports all the access specifiers like private, protected and public
d) **A struct cannot have protected access specifier**

69) What is the output of the program?

```
#include <iostream.h> main()
{
int a=5, b=10; if
   (a=b)
   cout<<"Hi";
   else
   cout<<"Hello
   ";
cout<<"Bye"<<a;
}
```

a) HiBye10 b) HelloBye10 c) Compilation Error d) HiBye5

70) What will happen to the following code?

```
#include <iostream.h> const
    int a=20;
void main()
{
 int *ptr;  const
    int* retA();
    ptr=retA();
 cout<<*ptr;
}
const int* retA()
{
 return &a;
}
```

a) warning **b) compilation error** c) neither warning nor compilation error

71) What will happen to the following code?

```
#include<iostream.h>
void main()
{
      int a=30;
      f();
}
void f()
{
      int b=30;
}
```

a) Successful output b) Warning **c) Compilation Error**

72) what is the output?

```
#include <stdio.h>
   float cal (float
   value)
{
return (3 * value);
}
void main()
{
int a = 10;
float b = cal ("123");
}
```

a) 369 b) 123

c) Compilation error - Cannot convert from char to float d) None of the above

73) What is the output of the program?

```
#include <iostream.h>
inline int max(int x, int y)
{
return(x > y ? x : y);
```

```
}
void main()
{
int(* max_func)(int,int)=max;
cout << max_func(75,33);
}
```

a) 75 b) Error - Undefined symbol max_func c) 33 d) None of the above

74) What is the output of the following?

```
#include <iostream.h> int
    add(int, int = 5, int = 10);
    void main()
                {
cout << add(10) << " " << add(10, 20) << " " << add(10, 20, 30);
}                                                                    int add(int a,
    int b, int c)
{
return a + b + c;
}
```

a) compilation error **b) 25 40 60** c) 15 30 60 d) 20 40 60

75) What will happen to the following code?

```
#include<iostream.h>
void main()
{
 int *ptr=new int;
    delete ptr;
    delete ptr;
}
```

a)Runtime Error b) Neither compilation nor Runtime Error c) Compilation Error

76) What will happen to the following code?

```
#include<iostream.h>
void main()
{
      int *ptr=new int;
      delete []ptr;
}
```

a)Runtime Error b) **Neither compilation nor Runtime Error** c) Compilation Error

77) What will happen to the following?

```
#include<iostream.h>
void accept(int x,int y)
{
 cout<<"in value method\n";
}
void accept(int &p,int &q)
{
 cout<<"in referece method\n";
}
void main()
```

```
{
 accept(45,55);
}
```

a) **output "in value method"** c) output "in referece method"
b) compilation error d) output "in value method in reference method"

78) What will happen to the following?

```
#include<iostream.h>
void main()
{
 cout<<30<<endl; int
    &ref=30; ref=60;
 cout<<ref<<endl;
}
```

a) output 30 30 **b) compilation error** c) output 30 60

79) What will happen to the following code?

```
#include<iostream.h> int
    num=200;
void main()
{
 int  const  *ptr;
    int*
    retNum();
    ptr=retNum();
 cout<<*ptr;
}
int* retNum()
{
 return &num;
}
```

a) **output 200** b) compilation error c) Runtime Error

80) What will happen to the following ?

```
#include<iostream.h>
void main()
{
 int val=300; int *
    const ptr;
    ptr=&val;
    *ptr=600;
    cout<<endl<<*ptr
    ;
}
```

a) compilation error c) output 600
b)output 300 d) output, garbage value

81) What is the output? #include<iostream.h>

```
void main()
{
 int num=20;
```

```
 void disp(int,int);
 disp(num,++num);
}
void disp(int a,int b)
{
 cout<<a<<"\t"<<b<<endl;
}
```

a) **1 21** b) 20 21 c) 20 20 d) 21 20

82) What will happen to the following program ?

```
#include<iostream.h>
  void main()
{
   int *ptr=new int;
  delete ptr;
   ptr=0;
   delete ptr;
  }
```

a) compilation error b) runtime error **c) neither compilation error nor runtime error**

83) What will happen to the following code?

```
#include<iostream.h>
int var=200;
void main()
{
 int& fun();  cout<<var<<endl;
  fun()=100;
 cout<<var<<endl;
}
int& fun()
{
 static int var=30;
 return var;
}
```

a) neither compilation error nor warning , output 200 100
b) warning
c) compilation error
d) **neither compilation error nor warning ,output 200 200**

84) what is the output ? #include<iostream.h> const int a=124;

```
void main()
{
 const int* sample();
 int * const  p=sample();
}
const int* sample()
{ return (&a);
}
```

a) **compile time error** b) runtime error c) neither compilation nor runtime error

85) What is the output?

```
#include<iostream.h>
    const int a=124;
    void main()
    {
     const int* sample();  int  const*  p;
     p=sample();
    }
    const int* sample()
    { return (&a);
}
```

a) compile time error b) runtime error **c) neither compilation nor runtime error**

86) Given

```
    #include<iostream.h>
    void disp()
    {
          int *ptr=new int;
    }
    void main()
    {
          disp();
}
```

In the above code after disp() method is over, the situation becomes

a) Dangling Poiner **b) Memory Leak** c) None of these

87) Given

```
    #include<iostream.h>
        void main()
        {
            int *ptr=new int;
        delete ptr;
              //Some other C++ Statements....
        }
```

In the above code after " delete ptr " statement, the situation becomes

a) Dangling Pointer b) Memory Leak c) None of these

88) What will happen #include <iostream.h>

```
  int a=20;
  void main()
  {
     int *ptr;  int *const retA();       ptr=retA();
     cout<<*ptr;
  }
   int *const retA()
  {
     return &a;
  }
```

a)neither compile ,nor runtime error b) runtime error c) compiletime error

89) What will happen #include <iostream.h>

```
const int a=20;
void main()
{
    int *ptr;  int *const retA();       ptr=retA();
    cout<<*ptr;
}
 int *const retA()
{
    return &a;
}
```

a) neither compile ,nor runtime error b) runtime error **c) compiletime error**

90) Will the following code work?

```
#include <iostream>
using namespace std;
int main ()
{
    int f()
    {
       return 10;
    }

    cout << f() << endl;
    return 0;
}
```

a) Yes **b) no**

91) Will the following code compile and link? Give reasons.

```
#include <iostream> using
    namespace std;
int main ()
{    int i = 0;
    int &ri(i);
   return 0;
}
```

a) **yes** b) no

92) Will the following code compile and link? Give reasons.

```
int main()
{    int
    i =
    0;
   int &ri = 0;

  return 0;
}
```

a) Yes **b) no**

93) Will the following code compile, link and execute?

==========

File a.h

```
int i;
==========================
File a.cpp '
int main ()
{
#include "a.h"
i = 0; return
    0;
}
```

a) **Yes** b) no

94) When the following two file, a.cpp and b.cpp are compiled, we get linking error. Why?
Compilation and linking command

```
cl.exe a.cpp b.cpp
File a.cpp
=======
int f(); int
    main()
    {
f();
return 0;
}
File b.cpp
=======
extern "C" int f();
int f()
{
return 0;
}
```

a) There is no main function inside "b.cpp"
b) **Function "f()" is declared but not defined inside "a.cpp"**
c) Function "f()" is declared with "extern" inside "b.cpp"
d) None of the above

95) What will be the output of the following program?

```
#include <iostream>
using namespace std;
int f()
{
cout << "f() called" << endl; return
    0;
}
int main ()
{
extern int f( );
return 0;
}
```

a) Output "f() called" b) Compiler error **c) No output** d) None of the above

96) Will the following code compile and link?

```
#define f main
int f()
{
return 0;
}
```

a) **Yes** b) no

97) What will be the output of the following code?

```
#include <iostream>
using namespace std; void
    f()
{
cout < < "First f function called" < < endl;
}
void f()
{
cout << "Second ffunction called" << endl;
}
Int main ()
{
 F();
 F();
 Return 0;
}
```

a) First function called second function called
b) First function called
c) Second function called
d) **Compiler error**

98) Is there anything wrong in the following code? If so, what?

```
int main ()
{ int x; x
    = x;
   return
   0;
}
```

a) Yes **b) No**

99) Is there anything wrong in the following code? If so, what?

```
int main ( )
{
const int x;
return 0;
}
```

a) Yes int cannot be made constant
b) No there is nothing wrong
c) **Yes const must be initialized**
d) None of the above

100) Will the following code compile and link?

```
    typedef int INT;
int main ()
{
INT i=0;
    return 0;
}
```

a) Yes b) no

101) What will be the output of the following program?

```
#include <iostream>
    using namespace
    std;
int main ()
{
Int i = 10; int
    *pi = &i;
    *pi = 100;
    cout << i
    << endl;
    return 0;
}
```

a) 10 **b) 100** c) Garbage d) None of these

102) What will happen to the following program?

```
#include <iostream>
    using namespace
    std;
int main ()
{ int i
    =
    20
    ;
const int *pi
    = &i; *pi =
    200; cout
    << i <<
    endl;
    return 0;
}
```

a) Compilation error b) Output 20 c) Output 200 d) None of these

103) What will happen ?

```
    #include<iostream.h>
void disp(int a=0,int b,int c)
{
cout<<a<<"\t"<<b<<"\t"<<c<<endl;
}
void main()
{
disp(10,20);
```

}

a) output 10 20 0 b) output 0 10 20 c) output 10 10 20 **d) error**

104) In case of function overloading

a) arguments must be different , return type may or may not be different
b) return type must be different,arguments may or may not be different
c) both return type and arguments must be same
d) both return type and arguments must be different

105) What will happen to the following code while compiling ? int& retVal()

```
{
int cnt=20;
return cnt;

}
```

a) No Error b) Error c) **Warning**

106)

```
#include<iostream.h>
void main()
{
char * const t="hello";
t="world";
}
```

a) Runtime Error b) **Compilation Error** c) Neither Compilation or Runtime Error

107)

```
#include<iostream.h>
int& disp()
{
int num=10;
  return num;
}
void main()
{
 disp()=30;
}
```

a) Compilation Error b) No Error,No Warning c) **Warning**

108)

```
#include<iostream.h> void main()
{
char *p="Hello";
  char *q=p;
  q="Good Bye";
  cout<<p<<"\t"<<q;
}
```

a) **Hello Good Bye** b) Good Bye Good Bye c) Error: Lvalue Reqd.

109)

```
#include<iostream.h>
const int a=124;
void main()
{
const int* sample();
  int *p;
```

```
    p=sample();
}
const int* sample()
   { return (&a);
}
```

a) Warning b) Neithe Warning nor Error c) **Compilation Error**

110)
```
#include<iostream>
void main()
{
   char t[]="String functions are simple";
     int len=strlen(t);
     cout<<len;
}
```

a) **Compilation Error** b) Warning c) successful output

111)
```
#include<iostream.h>
   void main()
    {
     int a=30;
     f();
    }
void f()
{
     int b=30;
}
```

a) Successful output b) Warning c) **Compilation Error**

112) What will happen to the following code?
```
#include<iostream.h>
     void main()
     {
     for(int i=0;i<5;i++)
     {       int a=0;
a++;
}
cout<<endl<<a;
}
```

a) **compilation error** b) it will print garbage value c) it will print 1 d) it will print 5

113) What will happen to the following code?
```
#include<iostream.h>
   void main()
{
for(int i=0;i<5;i++)
{
cout<<endl<<i;
}
for(int i=5;  ;i++)
{
cout<<endl<<i;
```

```
}
}
```

a) it will print 0 to 9
b) **infinite loop because there is no condition in second for loop**
c) compilation error

114) C++ compiler internally changes names of all functions at the declaration, definition and call. This process is known as _____ **or** _____

115) Default arguments can be given in the beginning or in between also.
a) True b) **False**

116) Function overloading and operator overloading comes under
a) Run time polymorphism b) **Compile time polymorphism**
c) Both a and b are correct d) None of the above

117) What will be the output of the following code?

```
#include<iostream.h>
#define MAXROW 3
   #define MAXCOL 4
   void main()
{
int (*p) [MAXCOL]; p=new
   int[MAXROW][MAXCOL];
   cout<<endl<<sizeof(p)<<endl<<sizeof(*p)
   ;
}
```

a) 2(under Dos) or 4(under Linux or windows) 8(under Dos) or 16(under Linux or windows)
b) 4(under Dos) or 8(under Linux or windows) 8(under Dos) or 16(under Linux or windows)
c) compilation error
d) runtime error

118) What is the output of the program?

```
#include <iostream.h>
void main ()
{
for(int j = 1, sum = 0; j < 5; j++)
sum += j; sum =
   j;
cout << sum;
}
```

a) 6 b) **5** c) Compilation error. Undefined variable sum and j d) 10

119) Which of the following is false about struct and class in C++?
a) The members of a struct are public by default, while in class, they are private by default
b) Struct and class are otherwise functionally equivalent
c) A class supports all the access specifiers like private, protected and public
d) A struct cannot have protected access specifier

120) What is the output of the program?

```
#include <iostream.h> main()
```

```
{
int a=5, b=10; if
    (a=b)
    cout<<"Hi";
    else
    cout<<"Hello";
cout<<"Bye"<<a;
}
```

a) HiBye10 b) HelloBye10 c) Compilation Error d) HiBye5 e) Bye10

121) What will happen to the following code?

```
#include <iostream.h> const
    int a=20;
void main()
{  int *ptr;  const
    int* retA();
    ptr=retA();
 cout<<*ptr;
}
const int* retA()
{
      return &a;
}
```

a) warning b) **compilation error** c) neither warning nor compilation error

122) What will happen to the following code?

```
#include<iostream.h>
void main()
{
      int a=30;
      f();
}
void f()
{
      int b=30;
}
```

a) Successful output b) Warning c) **Compilation Error**

123) what is the output?

```
    #include <stdio.h>
float cal (float value)
{
return (3 * value);
}
void main()
{
int a = 10;
float b = cal ("123");
}
```

a) 369 c) **Compilation error - Cannot convert from char to float**
b) 123 d) None of the above

124) What is the output of the program?

```
#include <iostream.h>
inline int max(int x, int y)
{
return(x > y ? x : y);
}
void main()
{
int(* max_func)(int,int)=max;
cout << max_func(75,33);
}
```

a) **75** b) Error - Undefined symbol max_func c) 33 d) None of the above

125) What is the output of the following?

```
#include <iostream.h> int
    add(int, int = 5, int = 10);
    void main()
                                    {
cout << add(10) << " " << add(10, 20) << " " << add(10, 20, 30);
}
     int add(int a, int b, int c)
{
return a + b + c;
}
```

a) compilation error b) **25 40 60** c) 15 30 60 d) 20 40 60

126) What will happen to the following code?

```
#include<iostream.h>
void main()
{
 int *ptr=new int;
    delete ptr;
    delete ptr;
}
```

a) **Runtime Error** b) Neither compilation nor Runtime Error c) Compilation Error

127) What will happen to the following code?

```
#include<iostream.h>
void main()
{
       int *ptr=new int;
       delete []ptr;
}
```

a) Runtime Error
b) **Neither compilation nor Runtime Error**
c) Compilation Error

128) What will happen to the following?

```
#include<iostream.h>
void main()
{
```

```
 cout<<30<<endl;  int
    &ref=30;  ref=60;
 cout<<ref<<endl;
}
```

a) output 30 30 b) **compilation error** c) output 30

129) What will happen to the following code?

```
#include<iostream.h
    > int num=200;
void main()
{
 int   const   *ptr;
    int*
    retNum();
    ptr=retNum();
      cout<<*ptr;
}
int* retNum()
{
      return &num;
}
```

a) output 200 b) **compilation error** c) Runtime Error

130) What will happen to the following ?

```
#include<iostream.h>
void main()
{
    int val=300;
      int * const
    ptr;
      ptr=&val;
      *ptr=600;
      cout<<endl<<*ptr;
}
```

a) **compilation error** b) output 600 c) output 300 d) output, garbage value

131) What is the output? #include<iostream.h>

```
void main()
{
 int num=20;
 void disp(int,int);
 disp(num,++num);
}
void disp(int a,int b)
{
 cout<<a<<"\t"<<b<<endl;
}
```

a) **21 21** b) 20 21 c) 20 20 d) 21 20

132) What will happen to the following program ?

```
      #include<iostream.h>
```

```
    void main()
{
   int *ptr=new
   int;        delete
   ptr;        ptr=0;
     delete ptr;
   }
```

a) compilation error b) **runtime error** c) neither compilation error nor runtime error

133) What will happen to the following code ?

```
#include<iostream.h>
int var=200;
void main()
{
 int& fun();  cout<<var<<endl;
   fun()=100;
 cout<<var<<endl;
}
int& fun()
{
 static int var=30;
 return var;
}
```

a) neither compilation error nor warning , output 200 100
b) warning
c) neither compilation error nor warning , output 200 200
d) compilation error

134) What is the output ?

```
   #include<iostream.h
   > const int a=124;
void main()
{
     const int* sample();
     int * const  p=sample();
}
const int* sample()
   {  return (&a);
}
```

a) **compile time error** b) runtime error c) neither compilation nor runtime error

135) what is the output ?

```
     #include<iostream.h>
const int a=124;
void main()
{
     const int* sample();
     int  const*  p;
     p=sample();
}
```

```
const int* sample()
   { return (&a);
}
```

a) compile time error b) runtime error c) **neither compilation nor runtime error**

136)

```
   Given
#include<iostream.h>
void disp()
{
      int *ptr=new int;
}
void main()
{
      disp();
}
```

In the above code after disp() method is over, the situation becomes

a) Dangling Poiner b) **Memory Leak** c) None of these

137) Given

```
#include<iostream.h>
     void main()
     {
          int *ptr=new int;
   delete ptr;
          //Some other C++ Statements....
     }
```

In the above code after " delete ptr " statement, the situation becomes

a) **Dangling Pointer** b) Memory Leak c) None of these

138) What will happen ?

```
   #include <iostream.h>
int a=20;
void main()
{
   int *ptr;  int
   *const retA();
     ptr=retA();
     cout<<*ptr;
}
 int *const retA()
{
      return &a;
}
```

a) **neither compile ,nor runtime error** b) runtime error c) compiletime error

139) what will happen?

```
   #include
   <iostream.h> const
   int a=20;
void main()
```

```
{
    int *ptr;  int
    *const retA();
      ptr=retA();
      cout<<*ptr;
}
 int *const retA()
{
      return &a;
}
```

a) neither compile ,nor runtime error b) runtime error c) **compiletime error**

140) What is the referent in the following code?

```
    int main ()
     {
int i = 0;
int &ri = i;

return 0;
}
```

a) ri **b) i** c) Both ri and i d)none

141) What is the output of the following code:

```
    #include <iostream>
    using namespace std;
    int main ()
   {
int x = 10,y= 20;
 if ( x > y );
  cout << "x is greater than y" < < endl;
      return 0;
    }
```

a) x is greater than y b) no output c) compiler error d) none of these

142) What is the output of the following code? Explain the reason.

```
    #include <iostream>
    using namespace std;
    int main()
   {
int i = 10;
                int j = 20;

int *pi = &i; int
    *pj = &j;

    if( pi = pj) {
cout << "Address of pi and pj are same" < < endl;
    }
    else {
cout << "Address of pi and pj are different" < < endl;
    }
```

```
    return 0;
}
```

a) address of pi and pj are same c) compiler erro
b) address of pi and pj are different d) none of these

143) What is the output of the following code:

```
int main()
{
   inti = 100;
   int &ri = i;

   ri = 200;
    ri = i;     i
    = ri;
    cout << i
    << endl;
    return 0;
}
```

a) 100 b) **200** c) 300 d)Compiler error

144) Write code in main function, which will output the value of the global variable i on the console.

```
#include <iostream>
using namespace std; int i =
    100;
int main()
{
int i = 500;
   // Write your code below this comment     return
    0;
}
```

a) cout<<i; **b) cout<<::i;** c) cout<<&i; d) You can't print global variable in main

145) What is the output in the following code:

```
#include <iostream >
using namespace std;
    int i = 100;
int& f()
{
    return i;
}
int main()
{
    f() = 200;
    cout << i <<
    endl;     return
    0;
}
```

a) 200 b)100 c) 300 d)Compiler error

146**)** What is the output of the following code:

```
#inc|ude <iostream>
    using namespace
    std;
int main()
{
const int j =
    100;    cout
    <<j << endl;
    j = 300;
    cout <<j <<
    endl;
    return 0;
}
```

a) 300 b) 100 c) 0 d) **Compiler error**

147) What is the output of the following code:

```
#include <iostream>
    using namespace
    std; int main()
{
int *pi;     *pi =
    100;     cout <<
    *pi << endl;
    return 0;
}
```

a) 100 b) Compiler error **c) Runtime error** d) 0

148) What is the output of the following code:

```
#include <iostream>
    using namespace
    std;
int main()
{
int a[3] = {10, 20, -
    30}; int *p = &a[1];
   P--;
   cout << *p << endl;
   P--;
   cout << p[3] << endl;
   return 0;
}
```

a) 10 garbage value **b) 10 -30** c) 10 20 d) Runtime error

149) what is the output?

```
    #include <iostream>
using namespace std;
void f( int i)
{
   i = 40;
}
void f1( int &k)
```

```
{
   k = 40;
}
int main()
{ int j = 0; cout <<
    j << endl;
f(j);
cout << j << endl;
f1(j );
cout <<j << endl; return
    0;
}
```

a) 0 40 40 b) 0 0 0 c) **0 0 40** d) Compiler error

150) What is the output?

```
    #include <iostream>
using namespace
    std; int i=0;
int& f()
{
    return i;
}
int g(int &ri)
{     ri
    =10
    0;
    return 0;
}
int main()
{
    cout << i << endl;
    g (f());
    cout << i << endl;
    return 0;
}
```

a) compilation error b) 0 0 c) 100 100 **d) 0 100**

151) What will happen? #include<iostream.h>

```
int val=100;
void main()
{
      int val=40;
    {
      int val=50;
      cout<<::val;
      }
}
```

a) output 100 b) output 50 c) output 40 d) compilation error

152) Will the following code compile and link? If not, give reasons for the error. int main ()

```
{ int i = (int)10;

return 0;
}
```

a) Yes b) No

153) Will the following code compile and link

```
int main ()
{
int i = 100,j = i;

return 0;
}
```

a) Yes b) No

154) Will the following code compile and link?

```
int main ()
{
int stdio = 0;
int iostream = 0;

return 0;
}
```

a) Yes b) no

155) What is the value of variable i after line 14:

```
01 int main ( )
02 {
03 inti = 10;
04
   05 i = 20;
06
   07  i = 10 + 30;
08
   09  i =40 + 0;
10
   11 i = 0 + 0;
12
13   i = 20;
   14  i +=
   5;
15
16 Return 0;

   }
```

a) 20 **b) 25** c) 0 d) 5

156) Will the following code compile and link?

```
int main ( )
{
virtual int j = 0;
```

```
return 0;
}
```
a) Yes b) **no**

157) In the following code, which variable will be created in stack memory? int i; int main ()
```
{
int j;

return 0;
}
```
a) I **b) j** c) both I and j d) none

158) Will the following code compile and link?
```
int main ()
{
int i;

&i;
return 0;
}
```
a) Yes b) no

159) Will the following code compile and link?
```
#define Begin
   { #define
   End     } int
   main ()
Begin
return 0;
End
```
a) Yes b) no

160) What kind of error we will get in the following code? Compilation Error or Linking Error?
```
void f();
int main ()
{
f();

return 0; -
}
```
a)compile time error b)**link error** c)runtime error d)successful execution

161) What is the value of the following on MS Windows 2000 or 32-bit implementation of Linux? sizeof (unsigned short int)

a) 2 bytes b) 3 bytes c) 4 bytes d) 8 bytes

162) What is the output from the following program?
```
#include <iostream>
using namespace std;
void f ()
```

```
{
Int i = 10; cout << i <
    < endl;
i++;
}
int main ()
{ f();
    f();
return 0;
}
```

a) 10 11 **b)10 10** c)Compiletime error d)None of the above

163) In the following code, function f returns a value which is an integer. In the function main, we are calling function f, but the return value we are not using or storing in any variable. Is this acceptable?

```
int f ()
{
return 100;
}

int main ()
}
      f();
return 0;
}
```

a) yes b) no

164) Will the following code give linking error as function f is not defined?

```
int f( );
int main ( )
{
return 0;
}
```

a) yes **b) no**

165) Will the following code compile and link? If yes, what will be the output of the following program?

```
#include <iostream> using
    namespace std;
#ifdef 0
int main()
{
cout << ”First main called" < < endl;
    return 0; #else int main()
{
cout << "Second main called" << endl; return
    0;
}
s
#endif
```

a) compiler error b) linking error c) successful output

166) What will happen to the following code?

```
#include<iostream> using
    namespace std;
#define Num #ifdef
    Num
int main()
{
cout << "First main called" << endl; return
    0;
}
#else int
    main()
    {
cout << "Second main called" << endl; return
    0;
}
#endif
```

a)compiler error b)**First main called**
c)Second main called d)None of the following

167) what will happen to the following code?

```
#include<iostream>
    using namespace
    std;
#ifdef Num
int main()
{
cout << "First main called" << endl;
    return 0;
}
#else
    int
    ma
    in()
{
cout << "Second main called" << endl;
    return 0;
}
#endif
```

a) compiler error as Num is not defined b)First main called
c)Second main called d)None of the following

168) What is the output from the following program?

```
#include <iostream>
using namespace std;
void f ()
{
 static int i = 10; cout
    << i < < endl;
i++;
}
int main ()
```

```
{ f();
    f()
    ;

return 0;
}
```

a) 10 11 b)10 10 c)Compiletime error d)None of the above

169) What is wrong in the following code?

```
int main ()
{

0 = 0; return
    0; }
```

a) nothing wrong **b) l-value error**

170) What is wrong in the following code? int main ()

```
{
      ;
return 0;
}
```

a) **nothing wrong** b) u cant have ; without any c++ expression

171) What is wrong in the following code? Will the following code compile and link?

```
int main ()
{
return 0; return
    1;
}
```

a) yes b) no

172) What is the output of the following code?

```
#include <iostream> using namespace std;
int main ()
{
int return = O; cout < < return << endl; return 0;
}
```

a) link error **b) compile error** c) runtime error d) successful output

173) What is the output of the following code?

```
#include <iostream>
    using namespace std,'
int main ()
{
int endl = 0; cout <<
    endl << endl;
    return 0;
}
```

a) 0 **b) 0 0** c) Compilation error d) Runtime error

174) What will happen to the following code?
```
int main ()
{
main( );
return 0;
}
```
a) Compile time error
b) Link error
c) **U need to terminate this program explicitly as recursion happens here**
d) None of the above

175) What will happen to the following code?
```
#define I  100
int main()
{ int i = I;
cout<<i<<endl; return 0;
}
```
a) 100 b) Garbage c)Compiler error

176) what will happen to the following code?
```
#define I  100 #undef I
int main()
{ int i = I;
cout<<i<<endl; return 0;
}
```
d)None of the following
a) 100 b) Garbage c) **Compiler error**

177) Will the following code compile?
```
int main ()
{ int int i;
return 0;
}
```
a) yes **b) no**

178) What is the output of the following program?
```
#include <iostream> using namespace std;
int main ()
{
cout << sizeof( int ) << endl; return 0;
}
```
d)None of the following
a) 4 b) 1 c) compilation error d) none of the above

179) Will the following program compile and link?
```
int main()
{
void v; return
    0;
}
```
a) yes **b) no**

180) What will be the output of the following code?
```
    #include <iostream>
using namespace std;
```

```
int main ()
{
cout << "Hi\n\tHello" < < endl; return
    0;
}
```

a) Hi and Hello on same line separated by tab
b) **Hi and Hello on different lines**
c) Compiler error as \n and \t can not be combined together
d) Hello

181) What will be the output of the following code?

```
#include <iostream> using
    namespace std;
int main ()
{
 int default = 0;

cout << default << endl; return
    0;
}
```

a) 0 b) compiler error : cannot give default as variable name
c) linking error d) runtime err

182) what is the output?

```
void printOutput(void); int
    main(void)
{
printOutput(); printOutput();
return 0;
}
void printOutput(void)
{
static int liVar = 102; liVar--;
printf("%d", liVar);
}
```

a) 101, 101 b) **101, 100** c) 102, 102 d) 102, 10

183) In the following C code snippet, what will be the output?

```
char *str = NULL;
if ((str != NULL) && (*str == 'A'))
{
printf("success\n");
}
else
{
printf("Not found\n");
}
```

(a) It can lead to a crash (b) Prints Success **(c) Prints not found** (d) Compile time error

184) Which of the following swap functions is correct (Swapping 2 int using pass bypointer approach)? a)

```
    void swap(int *x, int *y)
```

```
{
int *Z = 0;  *Z *
    *x;
*X = *y;
*Z * y ;
}
```

```
(b)   void swap(int *x, int *y)
{
int *Z = 0;
Z = *x;  X
    = Y; y
    = Z;
}
```

```
(c) Void swao(int *x, int *y)
{
int Z =0;
Z=*X;
*X=*y;
*y=Z;
```

```
(d)   Void swap(int x, int y)
{
int Z=0;
    Z=X;
    X=Y;
Y=Z;
}
```

185) Why does the following code give compilation error?

```
#include <iostream>

int main ()
{
cout << "main called" << endl; return
    0;
}
```

a) There is no "using namespace std" b) Iostream.h should have been there
c) #include <cout> is not there d) None of the above

186) In the following code iostream is a header file.

```
#include <iostream> using
    namespace std;
int main ()
{
cout << "main called" < < endl;
return 0;
}
```

a) True b) false

187) What will be the output from the following code?

```
#include <iostream> using
    namespace std;
int main ()
{
int i;
 cout << i << endl; return
    0;
}
```

a) 0 b) Garbage c) Compile error d) **Runtime error**

188) What does the following code do?

```
int main ()
{ int i (40);
return 0;
}
```

a) Assigning 40 to i
b) **Initializing I with 40**
c) Calling i function by passing 40
d) None of the above

189) Will the following code compile and link?

```
int main ()
{
int i = int(10);
return 0;
}
```

a) **Yes** b) no

190) Will the following code compile and link?

```
int main ()
{
int i = 100; int I
    = 200;


return 0;
}
```

a) Yes b) no

191) Will the following code compile and link?

```
int main ()
{
int i = 100; int j
    = i;
return 0;
}
```

a) Yes b) no

192) What is wrong in the following code? Will it compile and link?

```
int main ()
{{
return 0;
}}
```

a) It will compile but not linked
b) It will not compile
c) It will compile, link but fail at runtime.
d) **It will compile , link and run successfully.**

193) Which of the following statements are TRUE?
a) Reference variables must be initialized in C++
b) Array of reference is possible
c) Both A) and B)
d) None of the Above

194) What does extern "C" int Func(int *, short int); mean?
a) Declare Func as extern
b) **Will turn off "name mangling" for Func**
c) None of the above

195) Consider the following declarations in C enum colorslblack, blue, green };
This represent
a) black = 0, blue = 1, green = 2
b) color[l] = 'black', color[2] = 'blue', color[3] = 'green'
c) color = 'black' or color = 'blue' or color = 'green'
d) black = -1, blue = 0, green = 1;
e)Syntax error

196) What result is in the variable num after execution of the following statements?
int num = 58; num
%= 11;
a) **3** b) 5 c) 2 d) 1 1

197) What will be the output of this program?
#include <stdio.h>
int main(void)
{
int i = OX7;
i = i ^ i ;
printf("%d\n", i); return
0;
}
a) 1 b) 7 **c) O** d) 823543

198) Is the following C++ code safe?
int main(void)
{
char *szBuffer = new char[64]; strcpy(szBuffer,
"Financial Technologies"); szBuffer++;
delete [] szBuffer; return 0;
}
a) Yes b) **No**

199) What will be the output of the following?
int main(void)
{
int c = 7654; int
*pc = &c;
(*pC)++;

```
printf("%d, %d", (*pc), c); return
    0;
}
```

a) 7654, 7654 b) Some Address Value, 7655
c) Some Address Value, 7654 d**) 7655, 7655**

200) Are both of these code segments functionally same?

```
a) int *ptr = NULL;
b) int *ptr;
*ptr = NULL;
```

a)Yes **b) no**

201) When following piece of code is executed, what happens?

```
b=3; a =
    b++;
```

a) **a contains 3 and b contains 4** b) a contains 4 and b contains 4
c) a contains 4 and b contains 3 d) a contains 3 and b contains 3

202) What will happen? #include<iostream.h>

```
void main()
{
      disp();
}
void disp()
{
      cout<<"in disp";
}
```

a) warning **b) compilation error**
c) neither compilation nor warning d) runtime error

203) Malloc can call constructor, new can not call constructor. -
a) True b) **False**

204) Will the following C+ + program compile and link, or we need to include a header file like stdio.h or iostream? int main()

```
{
return 0;
}
```

a) It will compile but not linked b) It will not compile
c) It will compile, link but fail at runtim d) **It will compile, link and run successfully.**

205) Will the following C++ program compile and link, or we need to include a header file like stdio.h or iostream?

```
int main()
{
}
```

a) It will compile but not linked b) It will not compile
c) It will compile, link but fail at runtime. d) **It will compile , link and run successfully.**

206) What kind of error we will get in the following code? Compilation Error or LinkingError?

```
int main ()
```

```
{  0; return
    0;
}
```

a) It will compile but not linked
b) It will not compile
c) It will compile, link but fail at runtime.
d) It will compile , link and run successfully.

207) Will the following code compile and link? int main ()

```
{
10 + 5;
return 0;
}
```

a) It will compile but not linked
b) It will not compile
c) It will compile, link but fail at runtime.
d) It will compile , link and run successfully.

208) What kind of error we will get in the following code? Compilation Error on Linking Error?

```
int main ()
{
 i;
 return 0;
}
```

a) It will compile but not linked
b) **It will not compile**
c) It will compile, link but fail at runtime.
d) It will compile , link and run successfully.

209) What kind of error we will get in the following code? Compilation Error or Linking Error?

```
int main ()
{
i = 0; return
    0;
}
```

a) It will compile but not linked
b) It will not compile
c) It will compile, link but fail at runtime.
d) It will compile , link and run successfully.

210) Inline functions are replaced at
a) Run time b) **Compile time** c) Debug time d) None of above

211) Which type of variables can be referred from anywhere in the c++ code?
a) All variables
b) Universal variables
c) Local variables
d) Global variables

212) What is the value of sizeof(char)?
a) 1 b) 2 c) 4 d) 8

213)

214) If value has not type, then the pointer pointing to this value will be known as
a) Empty pointer b) Null pointer c) **Void pointer** d) None of above

215) Which arithmetic operation can be done in pointer?
a) Multiplication b) Division c) **Addition** c) None of above

216) Which operator is used for comparing two variables
a) := b) = c) =: d) ==

217) Can #define accept parameters
a) Yes b) No

218) What is the size of int datatype for 32 bit system?
a) 1 byte b) 2 byte c) **4 byte** d) 8 byte

219) How we define our name for constants?
a) #constant b) **#define** c) #define_constant d) #constant_define

220) \r is used for
a) carriage return b) new line c) end of the line d) vertical tab

221) C++ programs must contain
a) start() **b) main()** c) system() d) program()

222) Reference is like a
a) Pointer b)Structure c)Array d)None of above

223) Which is not C++ storage class?
a) auto b) register c) static d) **iostream**

224) What will happen ? #include<iostream.h>
```
void main()
{
int *ptr=new int;
      *ptr=30;
      cout<<endl<<*ptr<<endl;
}
```
a) compilation error b) runtime error c) warning **d) output : 30**

225) What will happen?
```
   #include<iostream.h>
void main()
{
for(int x=0;x<4;x++)
{
//some statements
}
for(int x=0;x<9;x++)
{
//some statements
}
}
```
a) warning **b) compilation error** c) neither warning nor compilation error

226) What will happen?
```
   #include<iostream.h>
void main()
{
for(int x=0;x<4;x++)
```

```
{
int j=4;
}
for(x=0;x<9;x++)
{
j++;
}
}
```

a) warning b) neither warning nor compilation error **c) compilation error**

227) Will following code work ?

```
#include<iostream.h>
void main()
{
const int num;
int const *ptr=&num;
}
```

a) No b) Yes

228) Will following code work ?

```
#include<iostream.h>
void main()
{
const int num=60;
int const *const ptr=&num;
}
```

a) No **b) Yes**

229) Will following code work ?

```
#include<iostream.h>
const int * fun()
{
static int num=40;
return &num;
}
void main()
{
int *ptr;
ptr=fun();
}
```

a) Yes **b) No**

230) Will Following code work?

```
#include<iostream.h>
const int * fun()
{
static int num=40;
return &num;
}
void main()
{
```

```
const int *ptr;
ptr=fun();
}
```

a) Yes b) No

231) Will following code work ?

```
   #include<iostream.h>
const int * fun()
{
static int num=40;
return &num;
}
void main()
{
int *const ptr=fun();
}
```

a) Yes **b) No**

232) Will following code work?

```
   #include<iostream.h>
int * const fun()
{
static int num=40;
return &num;
}
void main()
{
int * const ptr=fun();
}
```

a) Yes b) No

233) Will following code work ?

```
   #include<iostream.h>
int * const fun()
{
static int num=40;
return &num;
}
void main()
{
const int * ptr=fun();
}
```

a) No **b) Yes**

234) What will happen? #include<iostream.h>

```
void main()
{
int num=40;
   int &ref;
   ref=num;
ref++;
```

```
cout<<endl<<num;
}
```

a) error b) warning c) output 40 d) output 41

235) What will happen? #include<iostream.h>

```
void main()
{
const int num2=50;
int &ref=num2;
}
```

a) warning b) it will work c) **error**

236) What will happen?

```
#include<iostream.h>
void main()
{
int num2=50;
const int &ref=num2;
}
```

a) it will work b) error c) warning

237) Will following code work ?

```
#include<iostream.h>
void main()
{
int &ref=40;
}
```

a) No b) Yes

238) Will following code work ?

```
#include<iostream.h>
void main()
{
const int &k=400;
}
```

a) Yes b) No

239) What will happen?

```
#include<iostream.h>
int * const fun()
{
int num=40;
return &num;
}
void main()
{
const int * ptr=fun();
}
```

a. **warning** b) error c) neither warning nor compilation error

240) The new operator

a) Obtains memory for variable
b) **Returns the memory to the operating system**
c) Creates a variable called new
d) returns the information about currently available memory in the system

241) For the object for which it was called, a const member function (enhancement)
a) can modify both const and non-const member data
b) can modify only const member data
c) can modify only non-const member data
d) **can modify neither const or non-const member data**
e)

Oops

1) Copy Constructor is called when
a) Object is initialized using another object b) Object is assigned to another object
c) A and B both d) none of the above

2) What is the output ?

```
#include<iostream.h>
class myclass
{
public:  static
   int
   counter;
};
Int
   myclass::counter;
   void main()
{
 cout<<myclass::counter;
}
```

a) output 0 b) compilation error "static member must be initialized"
c) Linking error d) output garbage value

3) Use the following code to answer the question

```
Class Z {
  public:
    void def(char
a);          int
ghi();    private:
    char j;
      int
k;  };
```

Which of the following is legal in a program that uses this class, after the following declaration:
Z x;
a) x.ghi(); b) x.j = 'd'; c) Z.ghi(); d) None of the above is legal

4) How does a object refer to itself?
a) By passing itself to a constructor with itself as the parameter
b) There is no way for a class to refer to itself
c) By pointing to another class just like this one

d) **By using the this pointer**

5) Which of the following is not required in a class that contains dynamic allocation?
a) The copy constructor b) A constructor that copies variables into private variables
b) c) Destructor **d) All of the above are required**

6) What is the output ? #include<iostream.h>

```
class X
{  intnt j; public:  X()
 {
      this->j=0;
 }
 X(int n)
 {
      this->j=n;
 }
 X(const X &rhs)
 {
      this->j=rhs.j;
 }
};
void main()
{
 X
    x1,x2
    (5);  X
    x3(x2
    );
 x1=x3;
}
```

a. it will compile. Upon execution , the default constructor for 'X' will be called, then the overloaded constructor and then the copy constructor. The default assignment operator will be used.
b. It will fail during compilation because the copy constructor is attempting to use a const reference to modify a member variable.
c. It will compile. Upon execution, the default constructor for X will be called, then the overloaded constructor, and then a run-time error will occur when the assignment of x1=x3 is attempted.
d. It will compile. Upon execution, the default constructor for 'X' will be called once, and then the copy constructor will be called twice with last call being used to assign x1=x3.

7) Overloading is otherwise called as
a) virtual polymorphism **b) ad-hoc polymorphism**
c) transient polymorphism d) pseudo polymorphism.

8) Here is a function prototype and some possible function calls int day_of_week(int year,int month=1,int day=1);

```
//Possible function calls
Cout<<day_of_week();
Cout<<day_of_week(1995);
Cout<<day_of_week(1995,10);
Cout<<day_of_week(1995,10,4);
```

How many of the function calls are legal ?

a) 1 of them is legal
b) 2 of them is legal
c) 3 of them is legal
d) all of them are legal

9) Can we have a private constructor in a class ?
a) yes b) no c) no, only private functions are possible d) none of the above.

10) #include<iostream.h> class Alpha
{
public:
char data[10000];
Alpha();
~Alpha();
};
class Beta
{
public:
Beta(
)
{
n=0;
} void
FillData(Alpha
a); private: int
n;
};
How do u make the above sample code more efficient ?
a) if possible, make the constructor for Beta private to reduce the overhead of public constructors
b) change the return type in FillData to int to negate the implicit return conversion from "int" to "void"
c) make the destructor for Alpha virtual
d) pass a const reference to Alpha in FillData

11) What is the
output ?
#include<iostream.
h> class Sample
{
public:
int
*ptr;
Sampl
e(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;
}
void main()
{
 Sample s1=10;
    SomeFunc(s1);
 s1.PrintVal();
}
```

a) say I am in somefunc the value is 10 b) say I am in somefunc Null pointer
c) assignment (runtime error) **d) runtime error**

12) What is the output?

```
    #include<iostream.h>
class obj
{
public:
    obj(
    )
 {
      cout<<"in";
 }
 ~obj()
 {
      cout<<"out";
 }
};
void main()
{
 obj A,B;
 {
      obj D;
 }
    o
    b
    j
    E
    ;
}
```

a) in in in in out out out out **b) in in in out in out out out**
c) in in out out in in out out d) in in out out in out in out

13) What will be the output ?

```
#include<iostream.h>
    #include<string.h>
class A
```

```
{
 int code;
    char
    name[20
    ]; public:
    A()
 {
       code=0;
       strcpy(name,'\0');
 }
 A(int c,char *nm)
 {
       code=c;
       strcpy(name,nm);
 }
 A(A &obj)
 {
       code=obj.code;
       strcpy(name,obj.name);
}
void show();
};
void A::show()
{
 cout<<endl<<"code= "<<code<<endl<<"name="<<name;
}
void main()
{
 A
    obj1(20,"AAA
    ");  A
    obj2(obj1);
    obj1.show();
    obj2.show();
}
```

a) code=20 name= AAA for first and garbage value for second

b) code =20 name =AAA for both

c) Error: can not assign one object to another.

d) will not compile

14) What is the
output ?

```
#include<iostream.
h> class test
{
 int x;
    publi
    c:
 test(int y)
 {
       x=y;
```

```
}
int getX()
{
      int x=40;
      return this->x;
}
};
void main()
{
 test a(10);
 cout<<a.getX()<<endl;
}
```

compilation error

a) 10 b) 40 c) none of the above

15) What will happen

```
#include<iostream.
h> class name
{
public:
   name(
   )
{
      cout<<endl<<"in def con\n";
}
 name(name n)
{
cout<<endl<<"in copy con\n";
}
};
void main()
{
 name n1;
 name n2(n1);
}
```

a) output infinite "in copy con" b) output "in def const in copy con";

c) compile error d) run time error.

16) What will happen to the following code?

```
#include<iostream.h>
   class name
{
public:
 name(name &ref)
{
      cout<<endl<<"in copy con\n";
}
};
```

```
void main()
{
 name n1;
 name n2(n1);
}
```

a) output "in copy con" **b) compile error** c) linking error d) runtime error

17) What is the output ?

```
    #include<iostream.h>
class myclass
{
public:  static
    int
    counter;
};
Int
    myclass::counter;
    void main()
{
 cout<<myclass::counter;
}
```

a) output 0 b) compilation error "static member must be initialized"
c) Linking error d) output garbage value

18) What will happen to following code?

```
#include<iostream.h>
    class SomeClass
 {
    p
    u
    b
    li
    c:
SomeClass()
{
 cout<<endl<<"in SomeClass Def.Const\n";
 }
~SomeClass()
 {
cout<<endl<<"in SomeClass Destructor\n";

    }
    }
    ;
 void main()
 {
 SomeClass *s1=new SomeClass;
 }
```

a) output "in SomeClass Def.Const"
b) Runtime error because of memory leak.
c) output "in SomeClass Def.Const in SomeClass Destructor"

d) compilation error because of incorrect syntax of 'new'

19) What is the output ?

```
#include<iostream.
h> class myclass
{
public:

 static int counter;
};
void main()
{
 cout<<myclass::counter;
}
```

a) output 0 b) compilation error **c) Linking error** d) output garbage value

20) The copy constructor would take a parameter by reference only
a) True b) False

21) The default access scope for a method in a C++ class is
a) Private b) Public
c) Protected d) Default

22) Where does memory get allocated for a static data members of a class
a) Code/text b) Stack
c) Heap **d) Data**

23) Namespaces
a) Provide a logical grouping of objects
b) **Provide a logical grouping of classes**
c) Provide a physical grouping of objects
d) Provide a physical grouping of classes

24)
```
class Foo
{    int i;
};
```
In the above sample, what is the member access specifier of the member data "i"?
a) default b) virtual c) protected **d) private** e) public

25) A member function can always access the data
a) in the object of which it is a member b) **in the class of which it is a member**
c) in any object of the class of which it is a member d) in the public part of its class

26) which among the following type of pointer is used to represent an object that invokes a member function
a) void pointer b) null pointer c) **this pointer** d) base pointer

27) Which among the following operator is used to identify the class to which a member function belongs
a) Í b) [] c) :: d) .*

28) Which of the following is the default namespace of C++?
a) iostream b) standard **c) std** d) stdio

29) What operator is prepended onto the member function name to indicate that the function is a destructor?
a) & b) * **c) ~** d) :: e) –

30) Which one of the following statements is true about constructors and destructors?
a) Both explicitly declared constructors and explicitly declared destructors are required in a class.
b) Neither constructors nor destructors can take parameters.
c) In a given class, constructors are always required, but destructors are not.
d) **Constructors can take parameters, but destructors cannot.**
e) It is illegal to define either a constructor or a destructor as virtual

31) A const object can access only const function
a) true **b)** False

32) Select correct statement/s for destructor
a) Destructor is called when object goes out of scope
b) By default destructor is not provided by compiler
c) Destructor can not be overloaded
d) In case of inheritance base class destructor is called before derived class
e) Destructors can be virtual

33) Copy constructor is called in case…
a) When an object is initialized using another object
b) When object is passed to a function and collected in another object
c) When object is returned from a function and collected in another object
d) **All of the above**

34) What is the output?

```
#include<iostream.
h> class myclass
{
public:
 void myclass()
 {
      cout<<endl<<"in myclass def\n";
 }
 myclass(int k)
 {
      cout<<endl<<"in param const\n";
}
};
void main()
{
 myclass m1, m2(30);
}
```

a) output " in param const " b) output "in myclass def in param const"
c) compilation error d) runtime error

35) Argument of copy constructor is object of same class.
a) true **b) false**

36) copy constructor is called whenever object is initialized using another reference.
a) true b) false

37) What will happen to the following ?

```
#include <iostream.h>
class myclass
{
 static int
    cnt;
    public:
    static
    void
    disp()
 {
      cout<<this->cnt;
 }
};
void main()
{
 myclass::disp();
}
```

a) output 0 b) linker error c) output garbage value **d) compilation error**

38)

```
#include<iostream.h>
    class myclass
{
public:
 void myclass()
 {
      cout<<endl<<"in myclass def\n";
 }
 myclass(int k)
 {
      cout<<endl<<"in param const\n";
 }
};
void main()
{
 myclass  m2(30);
}
```

a) output " in param const " **b) compilation error** c) runtime error d) linker error

39) A _____ is a special member function used to initialize the data members of a class.

40) The default access for members of a class is _____.

41) Member functions of a class are normally made _____ and data members of a class are normally made _____.

42) The three member access specifiers are _____, _____ and _____.

43) _____ is called when we initialized one object using other object.

44) The size of a class with no data members and member functions is _____ byte.

45) _____ keyword if used , constructor will not be available for conversion.

46) Destructor can be overloaded.
a.true b.false

47) if the main function is coded as
[
mho a;
a=a-
a;
Then output will be
a) There was There was b) Nothing
c) There was a certain man There was a certain man. d) a run time error

48) if the
declaration mho
operator -
mho(y) Is
replaced by
mho operator –
mho(&y) And main
function is coded as
mho a; a=a-a;
Then the output will be
a) There was There was **b) There was There was a certain man**
c) There was a certain man There was a certain man d) compile time error

49) which of the following statement is correct?
a) **A constructor is called at the time of declaration of an object.**
b) A constructor is called at the time of use of an object.
c) A constructor is called at the time of declaration of a class.
d) A constructor is called at the time of use of a class.

50) Which one of the following options is correct?
a) Friend function can access public data members of the class.
b) Friend function can access protected data members of the class
c) friend function can access private data members of the class
d) **All of the above**

51) A copy constructor is invoked when
a) a function returns by value
b) **an object is passed by value to a function**

c) a function is returned by reference
d) an object is passed by reference to a function

52) The dot operator(or class member access operator) connects the following two entities reading from left to right

a) A class member and a class object
b) A class object and a class
c) A class and a member of that class
d) **A class object and a member of that class**

Operator Overloading

1) Operator= can be overloaded using
a) friend function **b) member function** c) both A and B d) none of the above

2) Which operators can be overloaded as non-member function?
a) () b) [] c) = **d) +**

3) Why is the extraction operator (>>) generally declared as a friend?
a) To allow the class to be read in a specific format.
b) To allow the operator to have access to private variables of the class
c) Since declaring the extraction operator part of the class will result in a
d) compilation error

4) In C++ programs the operation of the assignment operator and that of the copy constructor are
a) similar except that the copy constructor creates a new object
b) different except that they both copy member data.
c) **both (1) and (2)**
d) None of the above.

5) The next three questions are based on the following program segment

```
#include<iostream.h> class
    mho
{
public:
 mho(void)
 {
      cout<<"There was";
 }
 mho(mho &x)
      cout<<"a certain man";
 } {
 mho operator-(mho y)
 {
      mho ohm;
      return ohm;
 }
};
```

if the function main is coded as
mho a , b; then output will besss
a) There was There was b)Nothing

c) a runtime error d) There was a certain man There was a certain man.

6) which of the following operators cannot be overloaded ?
a) >> b) ++ **c) ?:** d) No such operator exists

7) What will happen ?

```
#include<iostream.h>
class opOverload
{
public:
    bool operator==(opOverload temp);
};
bool opOverload::operator==(opOverload temp)
{
    if(*this==temp)
    {
        cout<<"Both are same objects"<<endl;
        return true;
  }
    els
  e
    {
        cout<<"Both are different"<<endl;
        return false;
    }
}
void main()
{
  opOverload a1,a2;
    a1==a2;
}
```

a) compile time error b) Runtime error **c) No error**

8) What is the result ?

```
#include<iostream.h>
class myclass
{
private:
  int a,b; public:
    void set_ab(int
  i,int j)
    {
    a=i;
    b=j;
    }
    friend int sum(myclass);
};
int sum(myclass obj)
{
    return obj.a+obj.b;
```

```
}
void main()
{
   myclass c1,c2;
     c1.set_ab(10,20);
     c2.set_ab(40,40);
     cout<<endl<<sum(c1);
     cout<<endl<<sum(c2);
}
```

a) Error: can't access the member function without a reference to the class
b) Error: a non-member function can not access the data member of the class
c) **30 80**
d) Garbage value.

9) Which operators can not be overloaded using friend function?
a) () **b) =** **c) []** **d) ->**

10) virtual parent class is used for what
Ans: To solve "Diamond Problem" in hybrid inheritance

11) Which of the following statements is true?
a) Conversion operator function can have a void return type.
b) Conversion operator function must be written in destination
c) **Conversion operator function does not accept any argument**
d) Conversion operator function can be a friend function.

12) In which operator overloading, compiler implicitly passes zero as an argument ?
a) Post increment/decrement operator b) Pre increment/decrement operator
c) both pre and post d) subscript operator

13) In C++ programs the operation of the assignment operator and that of the copy constructor are
a) different except **similar except that the copy constructor creates a new object**
b) that they both copy member data.
c) both (1) and (2)
d) None of the above.

14) We can't do anything in source when converting from user defined to primitive type.
a) True **b) False**.

15) When you overload assignment operator using friend function 2 arguments are required.
a) true **b) false**

16) Which of the following statements is false ?
a) Conversion operator function must return a value
b) **Conversion operator function must be written in destination**
c) Conversion operator function does not accept any argument
d) Conversion operator function must be a member function.

17) Which of the following operators cannot be overloaded?
a) [] b) ++ c) **?:** d) *

18) When overloading a unary operator using a friend function
a) requires no argument
b) **requires one argument**
c) requires tow argument
d) take a defult argument

19) Operator function is declared in the
a) private section in the class
b) **public section in the class**
c) protected section in the class
d) Outside the class

Inheritance

1) What will happen to following code?

```
#include<iostream.h>
      class SomeClass
      {
   public:
      SomeClass()
      {
      cout<<endl<<"in SomeClass Def.Const\n";
      }
```

Consider the class inheritance:

```
      class B
      {
      public:
         B();
   B(int nn);
         void
   f();
   void g();
      private:

      int n;
      };
      class D: public B
      {
      public:
      D(int nn, float dd);
      void h();          private:
      double d;
      };
```

Which of the following functions can be invoked by an object of class D?
a) f()
b) g()
c) h()
d) All of the above

2) What will be the output ?

```
   #include<iostream.h>
class base
{
public:
      base()
      {
```

```
        cout<<"\nIn base const\n";
        print();
        }
        void disp()
    {
        print();
        }
        virtual void print()
        {
                cout<<endl<<"In base print\n";
        }
};
class derived:public base
{
public:
        derived()
        {
                cout<<endl<<"In derived const\n";
        }
        void print()
        {
                cout<<endl<<"In derived print\n";
        }
};
void main()
{
        derived d1;
        d1.disp();
}
```

a) In base const In derived const In base print In derived print
b) In base const In derived const In derived print In derived const
c) In base const In base print In derived print In derivd const
d) **In base const In base print In derived const in derived print**

3) What is true about c++ class and c++ struct
a) inheritance with c++ struct can be done
b) both can have member functions
c) c++ class members are private by default whereas c++ struct members are public by default
d) **all of the above**

4) Given the class declaration:
class D : public class B {/*…*/}
which of the following is true?
a) Public members of B become public members of D
b) Private members of D become public members of B
c) Protected members of B become public members of D
d) Private members of B become public members of D

5) If parent class has a method which is non-virtual, and child class defines the same method. It is called as

a) overloading b) overriding **c) redefinition** d) None of these.

6) Casting a base class pointer to derived class pointer is called as _____
a) Upcasting **b) Downcasting** c) abstraction d) None of the above.

7) When two or more objects are derived from a common base class, u can prevent multiple copies of the base class from being present in an object derived from those objects by declaring base class when it is inherited.
a) public b) protected **c) virtual** d) private

8)
```
#include<iostream.h>  class Base
{
public:
    int a;
    protecte
    d:
      int b;
    private:
      int c;
};
class Derived:Base
{
      int d;
      friend class Friend;
};
class Friend
{
      Derived derived;
};
```
In the above code, which of the following variables can be accessed in "Friend " ?
a) only a and b **b) a, b and d** c) only a d) error

9)
```
#include<iostream.h>
class A
{
    int a;
    public:
      void fun()
      {
      cout<<"from fun";
      }
};
class B:public A
{
    }
    ;
class C:virtual A
{
    }
    ;
class D:public B,C
```

```
{
   }
   ;
void main()
{
      D d;
      d.fun();
}
```

What will be the output of this program?

a) from fun **b) compile time error** c) run time error d) No output

10)
```
#include<iostream.h>
class base
{
public:
      base()
      {
            cout<<"\nbase def\n";
            base::disp();
      }
      void disp()
      {
            cout<<"base disp\n";
      }
};
class sub:public base
{
public:  sub()
 {
      cout<<"sub def\n";
      base::disp();
 }
 void disp()
 {
      cout<<"sub disp";
 }
};
void main()
{
 base *b=new base;
}
```

a) output " base def base disp" b) compilation error

c) output "base def base disp sub def sub disp" d) output " base def sub def base disp sub disp "

11) What is the output ?

```
   #include<iostream.h>
   class base
{
public:
      base()
      {
```

```
        cout<<"\nbase def\n";
    }
    void disp()
    {
        cout<<"base disp\n";
    }
};
class sub:public base
{
public:  sub()
 {
    cout<<"sub def\n";
    sub::disp();
 }
};
void main()
{
 sub s;
}
```

a) output "base def sub def"
b) compilation error
c) output "base def base disp sub def "
d) **output "base def sub def base disp "**
e) compilation error "disp not available in sub"

12)
```
#include<iostream.h>
  class base
{
public:
  base()
    {
    cout<<"\nbase def\n";
    sub::disp();
    }
    void disp()
    {
        cout<<"base disp\n";
    }
};
class sub:public base
{
public:
  sub()
    {
        cout<<"sub def\n";
    }
    void disp()
    {
        cout<<"sub disp\n";
    }
```

```
};

void main()
{
      sub s;
}
```

a) compilation error b) output "base def sub disp sub def"
c) output "in base def sub def sub disp " d) output "base def base disp sub disp"

```
13) #include <iostream.h>
    class base
{
public:  base()
 {
    cout<<"base def.\n";     disp();
 }
};
class sub:public base
{
public:  sub()
 {
      cout<<"sub def\n";
 }
 void disp()
 {
      cout<<endl<<"in sub disp\n";
 }
};
void main()
{

 base *b=new sub;
}
```

a) compilation error
b) output "in base def in sub def in sub disp"
c) output "in base def in sub disp in sub def"
d) output "in sub def in base def in sub disp"

14) When child class object is assigned to parent class object, object slicing takes place.
a) True **b) False**

15) Private members can be inherited but not accessible in derived class.
a) True **b) False**

```
16) #include <iostream.h>
    class base
{
public:  base()
 {
    cout<<"base def.\n";     disp();
```

```
 } void
    disp()
 {
       cout<<"\nbase disp\n";
 }
};
class sub:public base
{
public:  sub()
 {
       cout<<"sub def\n";
 } void
    disp()
 {
       cout<<endl<<"in sub disp\n";
 }
};
void main()
{
 base b=new sub;
}
```

a) compilation error
b) output "in sub def in base def in base disp"
c) output "in base def in sub def in sub disp"
d) output "in base def in base disp in sub def"

17) What is the output ?

```
    #include <iostream.h>
    class base
{
public:
    base()    {
      cout<<"base def.\n";
      disp();
      }
      void disp()
      {
            cout<<"\nbase disp\n";
      }
};
class sub:public base
{
public:
    sub()
      {
            cout<<"sub def\n";
      }
      void disp()
      {
            cout<<endl<<"in sub disp\n";
```

```
    }
};
void main()
{
    sub();
}
```

a) compilation error
b) output "in sub def in base def in base disp"
c) output "in base def in sub def in sub disp"
d) **output "in base def in base disp in sub def"**

18)
```
#include <iostream.h>
  class base
{
public:
    base()
    {
    cout<<"base def.\n";
    disp();
    }
    void disp()
    {
        cout<<"\nbase disp\n";
    }
};
class sub:public base
{
public:
 sub()
 {
    cout<<"sub def\n";
 void disp()
 {
 }
    cout<<endl<<"in sub disp\n";
 }
};
void main()
{
 base();
}
```

a) output "base def. base disp"
b) output "base def sub def sub disp "
c) output "base def sub def base disp"
d) compilation error " base() function not available "

19) When child class object is assigned to parent class object it is called as _____

20)
```
#include<iostream.h>
  class Base
{
```

```
    int static i;
    public:
      Base()
      {
      }
};
class Sub1:public virtual Base
{
    }
    ;
class Sub2:public Base
{
    }
    ;
class Multi:public Sub1,public Sub2
{
    }
    ;
void main()
{
      Multi m;
}
```

In the above program, how many times Base class constructor will be called ?

a) 1 **b) 2** **c)** 3 d) None

21) When two or more objects are derived from a common base class, u can prevent multiple copies of the base class from being present in an object derived from those objects by declaring base class when it is inherited.

a) public b) protected **c) virtual** d) private

22)
```
class A
    {
    public:
    A();        void ~A();
}
class B : public A { };
```

What is WRONG with the class declarations above?

a) Class B must explicitly define a constructor.
b) **The destructor in "A" cannot have a void return type.**
c) Nothing is wrong with the code above.
d) Class B must define a destructor
e) "A" must provide a copy constructor in order for it to be used as a base class.

23)
```
class X {    int  i;  protected:
float f;  public:   char  c;
};
class Y : protected X { };
```

Referring to the sample code above, which one of the following data members are accessible from class Y?

a) c only **b) f and c only** c) i and c only d) i and f only e) i, f, and c

24) class
IntArra
yRc :
public
IntArra
y;
What does the sequence of tokens ": public IntArray;" in the code above indicate?
a) It is the indicator that IntArray is derived from IntArrayRc class.
b) It is a scope resolution operator that states that IntArrayRc is a sub-class.
c) It is a scope resolution operator that states that IntArray is a super class.
d) **It is the indicator that IntArrayRc is derived from IntArray base class.**
e) It is the indicator for enforcing overloading of the IntArrayRc class from any IntArray class.

25) A class in C++ would be assumed as abstract if it has at least one virtual method
a) true **b) False**

26) What will be the output ?

```
#include <iostream.h>
class grandparent
{
public:
 grandparent(int k)
 {
      cout<<k<<endl;
 }
 grandparent()
 {
      cout<<0<<endl;
 }
};
class parent1:virtual grandparent
{
public:
 parent1(int j):grandparent(420)
 {
      cout<<j<<endl;
 }
};
class parent2:virtual grandparent
{
public:
 parent2(int j):grandparent(420)
 {
      cout<<j<<endl;
 }
};
class child:parent2,parent1
{
public:
 child(int m):parent1(100),parent2(200)
 {
```

```
        cout<<m<<endl;
 }
};
void main()
{
 child s(300);
} s
```

a) 420 100 200 300 b) 420 200 100 300 **c) 0 200 100 300** d) 0 420 200 100 300

27) What will be the output ?

```
   #include<iostream.h>
class base
{
public:  base()
 {
        cout<<"\nIn base const\n";
        print();
 }
 void print()
 {
        cout<<endl<<"In base print\n";
 }
};
class derived:public base
{
public:
derived()
        {
                cout<<endl<<"In derived const\n";
        }
        void print()
        {
                cout<<endl<<"In derived print\n";
        }
};
void main()
{
        derived d1;
}
```

a) In base const In derived const In derived print
b) In base const In derived print In derived const
c) In base const In base print In derived print In derivd const
d) **In base const In base print In derived const**

28) What will be the output ? #include <iostream.h>

```
class grandparent
{
public:
        grandparent(int k)
    {
        cout<<k<<endl;
```

```
    }
};
class parent1:virtual grandparent
{
public:
    parent1(int j):grandparent(420)
   {
    cout<<j<<endl;
    }
};
class parent2:virtual grandparent
{
public:
    parent2(int j):grandparent(420)
   {
    cout<<j<<endl;
    }
};
class child:parent2,parent1
{
public:
    child(int m):parent1(100),parent2(200)
    {
    cout<<m<<endl;
    }
};
void main()
{
    child s(300);
}
```

a) 420 100 200 300 b) 420 200 100 300

c) compilation error d) 0 420 200 100 300

29) A class is called as abstract base class if it has a _____function.

30) What is the output ?

```
  #include<iostream.h>
class professor
{
public:
    professor()
    {
        cout<<endl<<"professor";
    }
};
class researcher
{
public:
    researcher()
    {
```

```
            cout<<endl<<"researcher\n";
        }
};
class teacher:public professor
{
public:
        teacher()
        {
            cout<<endl<<"teacher";
        }
};
class myprofessor:public teacher,public virtual researcher
{
public:
        myprofessor()
        {
            cout<<endl<<"myprofessor\n";
        }
};
void main()
{
        myprofessor obj;
}
```

a) professor researcher teacher myprofessor **b) researcher professor teacher myprofessor**
c) myprofessor teacher researcher professor d)myprofessor researcher professor teacher

31) What is the order of execution of constructors in the hierarchy involving virtual base classes ?

a) i. virtual base class constructor , in the order of their inheritance
ii.non-virtual base class constructor, in the order of their inheritance iii. derived class constructor iv. constructors of member objects, in the order of their declaration.

b) i. virtual base class constructor, in the order of their inheritance
ii. derived class constructor. iii. constructors of member objects, in the order of their declaration iv. non-virtual base class constructor, in the order of their inheritance.

c) **i. virtual base class constructor, in the order of their inheritance**
ii. **non-virtual base class constructor, in the order of their inheritance iii. constructors of member objects, in the order of their declaration iv. derived class constructor**

d) i. derived class constructor
ii. constructors of member objects , in the order of their declaration
iii. non-virtual base class constructor, in the order of their inheritance iv. virtual base class constructor, in the order of their inheritance.

32) _____ enables reusability which saves time in development, and encourages using previouslyproven and high quality software.

33) A class which has pure virtual function is called as _____

34) When address of child class object is assigned to parent class pointer or reference, object slicing takes place.
a) True **b) False**

35) Protected members can be inherited but not accessible in derived class.
a) True **b) False**

36) In public inheritance mode protected and public members of parent class becomes _____ and _____ in child class respectively.

37) Which of the following access specifier is used as default in a class definition?
a) Protected b) Public **c) Private** d) Friend

38) How "Late binding" is implemented in C++?
a) Using C++ tables b) **Using Virtual tables**
c) Using Indexed virtual tables d) Using polymorphic tables

39) Which of the following is used to make an abstract class?
a) Declaring it abstract using static keyword.
b) Declaring it abstract using virtual keyword.
c) Making at least one member function as virtual function.
d) **Making at least one member function as pure virtual function.**

40) Which of the following statements is true about inheritance.
a) **If no constructors are specified for a derived class, objects of the derived class will use the constructors of base class.**
b) Private members of base class can be access by derived class member functions or objects of derived class.
c) An object of derived class can be access private or protected members of base class
d) In public inheritance the protected member of the base class become public for the functions outside the derived class

41) Which of the following cannot be used with the keyword virtual?
a) **class** b) member functions c) constructor d) destructor

42) Which of the following concepts is used to implement late binding?
a) **Virtual function** b) Operator function c) Const function d) Static function

43) Which inheritance type is used in the class given below?

```
Class A :public X, public Y
{}
```

a) Multilevel inheritance b) **Multiple** inheritance
c) Hybrid inheritance d) Hierarchical inheritance

44) Assume a class Derv that is privately derived from class Base. An object of class Derv located in main() can access
a) **public member of Derv** b) protected members of derv
c) public member of Base d) protected members of Base

45) When both base and derived class contain constructors & destructor's which of the following choice is correct

a) Both constructors & Destructors are executed in reverse order of derivation.
b) Both Constructors & Destructors are executed in their order of derivation.
c) **Constructors are executed in their order of derivation and Destructors are executed in the reverse order of derivation**
d) Constructors are executed in reverse order of derivation and Destructors are executed in their order of derivation

Late Binding

1) include<iostream.h>
class myclass
{
public:
virtual void
f2()
{
cout<<endl<<"in f2\n";
}
virtual void f1()
{
cout<<endl<<"in f1\n";
}
void fun()
{
int *ptr=(int*)this;
ptr=(int *)*ptr;
ptr=(int*)*ptr;
}
};
void main()
{
myclass m;
m.fun();
}
when fun() function is over, what does ptr stores ?
a) address of virtual poiner b) address of f1 **c) address of f2** d) none of the above

2) What is the output ?
#include<iostream.h>
class base
{
public:
virtual void disp()
{
cout<<"base disp\n";
}
};
class sub1:public base

```
{
public:
	void disp()
	{
		cout<<"sub1 disp\n";
	}
};
class sub2:public sub1
{
public:
	void disp()
	{
		cout<<endl<<"sub2 disp\n";
	}
};
void main()
{
   base *b;
     sub1
   s1,*s2;
     sub2
   s3,*s4;

     b=new base;
     s2=dynamic_cast<sub1*>(b);
     if(s2)
     {
s2->disp();
}
   else
     {
		cout<<"failed\n";
     }
b=&s3;
     s4=dynamic_cast<sub2*>(b);
     if(s4)
     {
		s4->disp();
   }		else	{
		cout<<"failed\n";
     }
   }
```

a) sub1 disp sub2 disp b) compilation error c) sub2 disp sub2 disp **d) failed sub2 disp**

3) Which of the following can be virtual?
a) constructors **b) destructors** c) static functions d) None of the above

4) VTABLE contains
a) **addresses of virtual functions** b) addresses of virtual pointers
c) address of virtual table d) None of the above

5) What will be the output ? #include<iostream.h>

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

a) 00000
1010

b) 01101
01010

c) 01011
11010

d) 10100
11011

6) What is the output ?

```
   #include<iostream.h>
class base
{
```

```
public:
      virtual void
   f1()
     {
     }
};
class sub:public base
{
   }
   ;
void main()
{
      sub s;
      cout<<sizeof(s)<<endl;
}
```

a) 0 b) 1 size of empty class is always 1 **c) 4** d) 5

7) What is the output ?

```
   #include<iostream.h>
class base
{
public:       virtual
   void disp()
     {
              cout<<"base disp\n";
     }
};
class sub1:public base
{
public:
      void disp()
      {
              cout<<"sub1 disp\n";
      }
};
class sub2:public sub1
{
public:
      void disp()
      {
              cout<<endl<<"sub2 disp\n";
      }
};
void main()
{
   base *b;
     sub1
   s1,*s2;
     sub2
   s3,*s4;
```

```
    b=&s1;
    s2=dynamic_cast<sub1*>(b);
    if(s2)
    {
        s2->disp();
  }
    els
  e
    {
        cout<<"failed\n";
    }
    s4=dynamic_cast<sub2*>(b);
    if(s4)
    {
        s4->disp();
  }
    els
  e
    {
        cout<<"failed\n";
    }
}
```

a) Error b) sub1 disp sub2 disp

c) sub1 disp failed d) sub1 disp sub1 disp e) sub2 disp sub2 disp

8) What is the output ?

```
  #include<iostream.h>
class base
{
public:
    virtual void disp()
    {
        cout<<"base disp\n";
    }
};
class sub1:public base
{
public:
    void disp()
    {
        cout<<"sub1 disp\n";
    }
};
class sub2:public sub1
{
public:
    void disp()
    {
        cout<<endl<<"sub2 disp\n";
    }
};
```

```
void main()
{
    base *b;
      sub1
    s1,*s2;
      sub2
    s3,*s4;

      b=&s3;
      s2=dynamic_cast<sub1*>(b);
      if(s2)
      {
              s2->disp();
    }
      els
    e
      {
              cout<<"failed\n";
      }
      s4=dynamic_cast<sub2*>(b);
      if(s4)
      {
              s4->disp();
    }
      els
    e
      {
              cout<<"failed\n";
      }
}
```

a) sub2 disp sub2 disp b) sub1 disp sub2 disp c) failed sub2 disp
d) compilation error e) sub2 disp failed

9) Given the following code :

```
    #include<iostream.h>
class base
{
public:
    virtual
    void disp()
 {
      cout<<endl<<"in base disp\n";
 }

};
class sub1:public base
{
public:  void disp()
 {

 }
```

```
 void print1()
 {
    cout<<endl<<"in print1\n";
 }
};
void main()
{
 base *b;  sub1 s1,*s2,*s3;
b=new base;  s2=static_cast<sub1*>(b);
s3=dynamic_cast<sub1*>(b);   c
out<<s2<<endl;
 cout<<s3<<endl;
}
```

a) s2 will contain NULL, s3 not null **b) s3 will contain NULL, s2 not null**
c) both will contain NULL d) both will contain Not NULL

10) What will be the output ? #include<iostream.h>

```
class base
{
public:  virtual void
    disp()=0;  base()
 {
      disp();
 }
};
class sub:public base
{
public:
 void disp()
 {
      cout<<endl<<"in sub disp\n";
 }
};
void main()
{
 base *b=new sub;
}
```

a) compilation error b) output "in sub disp" **c) linking error** d) runtime error

11) What will be the output ?

```
   #include<iostream.h>
class base
{
public:
      virtual void disp()
      {
            cout<<endl<<"in base disp\n";
      }
};
class sub:public base
{
```

```
public:
        void disp()
        {
                cout<<endl<<"in sub disp\n";
        }
        void print()
        {
                cout<<endl<<"in print";
        }
};
void main()
{
    base *b=new sub;
      b->disp();        b-
    >print();
}
```

a) output "in base disp in print" b)output "in sub disp in print"

c) compilation error d) output "in sub disp in base disp in print"

12)
```
#include<iostream.h>
    class myclass
{
public:
        virtual void f2()
        {
                cout<<endl<<"in f2\n";
        }
        virtual void f1()
        {
                cout<<endl<<"in f1\n";
        }
        void fun()
        {
                int *ptr=(int*)this;
        ptr=(int *)*ptr;
        ptr++;
                ptr=(int*)*ptr;
        }
};
void main()
{
    myclass m;        m.fun();
}
```

when fun() function is over, what does ptr stores ?

a) address of virtual poiner **b) address of f1** c) address of f2 d) none of the above

13) Given the following code :

```
    #include<iostream.h>
class base
{
public:
```

```
virtual void disp()
{
      cout<<endl<<"in base disp\n";
    }
};
class sub1:public base
{
public:
 void disp()
 {
      cout<<endl<<"in sub1 disp\n";
 }
 void print1()
 {
            cout<<endl<<"in print1\n";
      }
};
void main()
{
      base *b;
sub1 s1,*s2,*s3;
   b=new base;
     s2=static_cast<sub1*>(b);
   s3=dynamic_cast<sub1*>(b);
     cout<<s2<<endl;
     cout<<s3<<endl;
}
```

a) s2 will contain NULL, s3 not null **b) s3 will contain NULL, s2 not null**
c) both will contain NULL d) both will contain Not NULL

13) The operator used for getting the type_info object is
a) Typeof **b) Typeid** c) Type d) Typeinf

15) All method invocations in C++ by default exhibit late binding
a)True b) **False**

16) To get polymorphism for a class you have to mark your methods as
a) Static b) **Virtual** c) Pure virtual d) Final

17) If a dynamic cast fails
a) aIt throws an exception b) **Returns a null value**
c) Converts to desired type d) Can never say

18) A constructor can be marked as virtual
a)True **b) False**

19) What is the output ? #include <iostream.h>

```
class base
{
public:  base()
```

```
{
      cout<<"base def.\n";
      disp();
}
virtual void disp()=0;
};
class sub:public base
{
public:  sub()
{
      cout<<"sub def\n";
}
void disp()
{
      cout<<endl<<"in sub disp\n";
}
};
void main()
{

      base *b=new sub;
}
```

a**) linker error** b) compilation error
c) output "in base def in sub def in sub disp" d) runtime error

20) #include<iostream.h> class first
```
{
      int a;
      virtual void fun(){}
};
```
What is the size of the class ? (assume 16 bit architecture)
a) 1 byte b) 2 byte c) 3 byte d) **4 byte**

21) Virtual pointer (vptr) is initialized inside virtual function
a) True b) **False**

22) If a class has 5 virtual functions, then 5 virtual tables will be created.
a)True b) **False**

23) There is only one virtual table gets created per object.
a)True b) **False**

24) In case of virtual functions all the objects of a class share virtual pointer.
a)True b) **False**

25) #include <iostream.h> class base
```
{ public:
      base()
      {
            cout<<"in base def.\n";
            disp();
```

```
        }
        virtual void disp()
        {
                cout<<endl<<"in base disp\n";
        }
};
class sub:public base
{ public:
        sub()
        {
                cout<<"in sub def\n";
        }
        void disp()
{
                cout<<endl<<"in sub disp\n";
        }
};
void main()
    {
       base *b=new sub;
}
```

a) output " base def sub def in sub disp"
b) compilation error
c) output "in base def in base disp sub def in sub disp"
d) **output "in base def in base disp in sub def"**

```
26)  #include <iostream.h>
    class base
{
public:  base()
 {
    cout<<"base def.\n";     disp();
 }
 virtual void disp()=0;

};
class sub:public base
{
public:  sub()
 {
        cout<<"sub def\n";
 }
 void disp()
 {
        cout<<endl<<"in sub disp\n";
 }
};
void main()
{
```

```
 base *b=new sub;
}
```

a) linker error b) compilation error
c) output "in base def in sub def in sub disp" d) runtime error

14) In case of dynamic polymorphism, availability of child class object in a base pointer can be checked using either _____ or _____.

15) Virtual pointer (vptr) points to virtual function.
a)true **b)false**

16) There is only one virtual table gets created for a class no matter how many instances are created.
a) true **b) false**

17) Abstract class can not have non-virtual functions.
a)true **b) false**

File Handling

1) Difference between text and binary mode is based on
a) How newline is treated b) How End Of File is represented
c) How numeric data is stored **d) all of the above**

2) What is false about cin?
a) object of istream b) represents standard input
c) it is not a function **d) it is used to read input from user's terminal**

3) 'ios' stream is derived from iostream
a) true b) false

4) The objects that correspond to the standard devices on the system include
a) cin b) cout c) clog **d) All of the above.**

5) Which of the following is the base class of C++ steam class hierarchy?
a) istream b) iostream c) stream **d) ios** e) ostream

6) Serialization is the process of
a) Converting bytes to objects **b) Converting objects to bytes**
c) Converting bytes to classes d)Converting classes to bytes

7) Which is the proper prototype for overloading the ">>" operator for a class like Cpoint
a) istream operator>>(istream, CPoint); b) istream operator>>(istream&, CPoint);
c) istream& operator>>(istream&, CPoint); **d) istream& operator>>(istream&, CPoint&)**

8) extraction operator is used with cout.
a) True **b) False**

9) The class which allows us to read as well as write in a file is _____.

Templates

1) Templates can be distributed to the client through
a) **header file** b) lib file c) both A and B d) templates can not be distributed at all

2) Which of the following is not a valid initialization of a template class, assuming the class is declared as follows:

```
template<class T>
class Pair { }
```

a) Pair <int>
b) Pair<char>
c) Pair <abc> (assuming abc is a user defined class)
d)All of the above are valid initializations of a template class

3) The STL makes abundant use of
a) inheritance b) virtual functions
c) friend functions **d) None of the above.**

4) Template classes can be inherited
a) True b) False

5)
```
#include<iostream.h>  template<class T,class X>
class obj
{
   T my_t;   X my_x;
   public:
   {
   obj(T t,X x):my_t(t),my_x(x)
    }
      };
```

Referring to the sample code above which one of the following is a valid conversion operator for the type T ?
a) T operator T(){ return my_t;} b) T operator (T) const{return my_t;}
c) operator(T) {return my_t;} **d) operator T() const{ return my_t;}**

6) Given following class template.

```
#include <iostream.h>
{
 template<class t1,class t2> class myclass
};
```

Write a statement which will direct a compiler to
a) generate this class for double and char respectively.
b) Create object of this class "m1" on stack.

7) Which one support unknown data types in a single framework ?
a) inheritance b)virtual functions c) abstract base class **d) templates.**

8) Which one support unknown data types in a single framework ?
a) inheritance b) virtual functions c) abstract base class d) **templates**

9) A vector is an appropriate container if you
a) Want to insert lots of new elements at arbitrary locations.
b) Want to insert new elements, but always at the front of the container.
c) **Are given an index number and you want to quickly access the corresponding element.**
d) Are given an element's key value and you want to quickly access the corresponding elements.

10) An STL algorithm is
a) **A standalone function that operates on containers.**
b) A Link between member functions and containers.
c) a friend function of appropriate container classes.
d) a member function of appropriate container classes.

11) Actual code for template function is generated when
a) the function declaration appear in the source code
b) the function definition appears in the source code
c) **a call to the function appears in the source code**
d) the function is executed at runtime

12) Which of the following statement about template is not correct
a) The compiler generates only one version of function template for each data type irrespective of the number of calls that are made for that type
b) A function template can have multiple arguments
c) **Using templates saves memory**
d) We can inherit a new class from the class template

13) Which among the following is an associative container
a) vector b) deque c) **set** d) stack

Exception

1. What happens to the automatic objects that have been constructed in a try block when that block throws an exception ?
a) only throws exception
b) **Destructors are called for each of the objects**
c) same as for other variables.
d) None of the above.

2. Exceptions are thrown
a) from the catch block to the try block
b) from a throw statement to the try block
c) from the point of error to a catch block
d) **from a throw statement to a catch block**