**C++ Moderate**

1. What is virtual inheritance?
2. C++ technique to avoid multiple copies of the base class into children/derived class
3. C++ technique to avoid multiple inheritances of classes
4. C++ technique to enhance multiple inheritance
5. C++ technique to ensure that a private member of the base class can be accessed somehow

Answer: a  
Explanation: Virtual inheritance is a C++ technique with which it ensures that a derived class contains only one copy of the base class’s variables.

1. Overloaded functions are
   * 1. Very long functions that can hardly run
     2. One function containing another one or more functions inside it
     3. Two or more functions with the same name but different number of parameters or type
     4. None of the mentioned

Answer: c  
Explanation: This is the definition of function overloading i.e. function having same name but different number of parameters and types.

1. How to handle the exception in constructor?

a) We have to throw an exception  
b) We have to return the exception  
c) We have to throw an exception & return the exception  
d) none of the mentioned

Answer: a  
Explanation: As a constructor don’t have a return type, We have to throw the exception.

1. Which one is suitable syntax for function template?
   1. template< class T> return\_type Function\_Name(parameters)
   2. template< typename T> return\_type Function\_Name(parameters)
   3. both a and b
   4. None of these

Answer: c

1. Which of the followings are false statements about Local class?  
     
   1. A local class type name can only be used in the enclosing function  
   2. All the methods of Local classes must be defined inside the class only  
   3. A Local class can contain static data members.   
   4. A Local class may contain static functions.  
   5. Non-static variables of the enclosing function are not accessible inside local classes.   
   6. Local classes cannot access global types, variables and functions**.**
   1. Only 1,3
   2. Only 3, 6
   3. Only 2 , 4 , 6
   4. None of these

Answer: b

1. Default value of static variable is\_\_\_\_\_.
   * 1. 0
     2. 1
     3. Garbage value
     4. Compiler dependent

Answer: a;

1. Predict the output of this program

#include<iostream>

using namespace std;

class Test

{

private:

  int x;

  int y;

public:

  Test(int x = 0, int y = 0) { this->x = x; this->y = y; }

  static void fun1() { cout << "Inside fun1()"; }

  static void fun2() { cout << "Inside fun2()"; this->fun1(); }

};

int main()

{

  Test obj;

  obj.fun2();

  return 0;

}

1. Inside fun2() Inside fun()1
2. Inside fun2()
3. Inside fun1() Inside fun2()
4. Compile error.

Answer: d;

There is error in fun2(). It is a static function and tries to access this pointer. this pointer is not available to static member functions as static member function can be called without any object.

1. Predict the output

#include<iostream>

using namespace std;

class Base {};

class Derived: public Base {};

int main()

{

    Base \*bp = new Derived;

    Derived \*dp = new Base;

}

1. No Compile error.
2. Compile error in line “Base \*bp = new Derived;”
3. Compile error in line “Base \*bp = new Derived;”
4. Runtime error.

Answer: c;

A Base class pointer/reference can point/refer to a derived class object, but the other way is not possible.