1. What is meant by multiple inheritance?

A	Deriving a base class from derived class
B	Deriving a derived class from base class
C	Deriving a derived class from more than one base class
D	None of the mentioned

2. Which symbol is used to create multiple inheritance?

A	Dot
B	Comma
C	Dollar
D	None of the mentioned

3. Which of the following advantages we lose by using multiple inheritance?

A	Dynamic binding
B	Polymorphism
C	Both a and b
D	None of the mentioned

4. In …… inheritance, the constructors are executed in the order of inheritance.

5. What is the output of this program?

```
#include <iostream>
    using namespace std;
    class polygon
    {
        protected:
        int width, height;
        public:
        void set_values (int a, int b)
        {
            width = a; height = b;}
        };
        class output1
        {
            public:
                void output (int i);
        };
    void output1::output (int i)
    {
        cout << i << endl;
    }
    class rectangle: public polygon, public output1
    {
        public:
        int area ()
        {
            return (width * height);
        }
    };
    class triangle: public polygon, public output1
    {
        public:
        int area ()
        {
            return (width * height / 2);
        }
    };
    int main ()
    {
        rectangle rect;
        triangle trgl;
        rect.set_values (4, 5);
        trgl.set_values (4, 5);
        rect.output (rect.area());
        trgl.output (trgl.area());
        return 0;
    }
```

A	20
B	10
C	20 10
D	None of the mentioned

6. What is the output of this program?

```
#include <iostream>
    using namespace std;
    class Base
    {
        public:
        virtual void print() const = 0;
    };
    class DerivedOne : public Base
    {
        public:
        void print() const
        {
            cout << "DerivedOne\n";
        }
    };
    class DerivedTwo : public Base
    {
        public:
        void print() const
        {
            cout << "DerivedTwo\n";
        }
    };
    class Multiple : public DerivedOne, public DerivedTwo
    {
        public:
        void print() const
        {
            DerivedTwo :: print();
        }
    };
    int main()
    {
        int i;
        Multiple both;
        DerivedOne one;
        DerivedTwo two;
        Base *array[ 3 ];
        array[ 0 ] = &both;
        array[ 1 ] = &one;
        array[ 2 ] = &two;
        array[ i ] -> print();
        return 0;
    }
```

A	DerivedOne
B	DerivedTwo
C	Error
D	None of the mentioned

```
#include <iostream>
    using namespace std;
    class student
    {
        public:
        int rno , m1 , m2 ;
        void get()
        {
            rno = 15, m1 = 10, m2 = 10;
        }
    };
    class sports
    {
        public:
        int sm;
        void getsm()
        {
            sm = 10;
        }
    };
    class statement:public student,public sports
    {
        int tot,avg;
        public:
        void display()
        {
            tot = (m1 + m2 + sm);
            avg = tot / 3;
            cout << tot;
            cout << avg;
        }
    };
    int main()
    {
        statement obj;
        obj.get();
        obj.getsm();
        obj.display();
    }
```

A 3100

B 3010

C 2010

D 1010

8. What is the output of this program?

```
#include <iostream>
    using namespace std;
    struct a
    {
        int count;
    };
    struct b
    {
        int* value;
    };
    struct c : public a, public b
    {
    };
    int main()
    {
        c* p = new c;
        p->value = 0;
        cout << "Inherited";
        return 0;
    }
```

A Inherited

B Error

C Runtime Error

D None of the mentioned

9. What is the output of this program?

```
#include <iostream>
    using namespace std;
    class Base1
    {
        protected:
        int SampleDataOne;
        public:
        Base1()
        {
            SampleDataOne = 100;
        }
        ~Base1()
        {
        }
        int SampleFunctOne()
        {
            return SampleDataOne;
        }
    };
    class Base2
    {
        protected:
        int SampleDataTwo;
        public:
        Base2()
        {
            SampleDataTwo = 200;
        }
        ~Base2()
        {
        }
        int SampleFunctTwo()
        {
            return SampleDataTwo;
        }
    };
    class Derived1 : public Base1, public Base2
    {
        int MyData;
        public:
        Derived1()
        {
            MyData = 300;
        }
        ~Derived1()
        {
        }
        int MyFunct()
        {
            return (MyData + SampleDataOne + SampleDataTwo);
        }
    };
    int main()
    {
        Base1 SampleObjOne;
        Base2 SampleObjTwo;
        Derived1 SampleObjThree;
        cout << SampleObjThree.Base1 :: SampleFunctOne() << endl;
        cout << SampleObjThree.Base2 :: SampleFunctTwo() << endl;
        return 0;
    }
```

A	100
B	200
C	Both a and b
D	None of the mentioned