Programming in Modern C++: Assignment Week 5

Total Marks : 25

Partha Pratim Das
Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur, Kharagpur – 721302
partha.p.das@gmail.com

February 14, 2025

Question 1

Consider the following code segment. *[MCQ, Marks 2]*

```
#include <iostream>
using namespace std;
class myClass1 {
    static int x1;
    double x2;
    public:
        void fun() { cout << "fun" << endl; }
};
class myClass2 : public myClass1 {
    double d1;
};
int myClass1::x1 = 0;
int main(){
    myClass2 d;
    cout << sizeof(d);
    return 0;
}
```

What will be the output? (Assume `sizeof(double) = 8` and `sizeof(int) = 4`)

a) `16`

b) `12`

c) `20`

d) `24`

Answer: a)
Explanation:
`static` member doesn't take part in inheritance. So, derived class will not inherit base class data member `x1`. Only `x2` will be inherited from class `myClass1`. So, the size of object `d` will be 8 bytes.

Question 2

Consider the following code segment. *[MCQ, Marks 2]*

```
#include <iostream>
using namespace std;
class base {
protected:
    int n1;
    public:
        base(int x) : n1(x) { }
        void f1() { cout << ++n1 << endl; }
};
class derived : public base {
    public:
        derived(int x) : base(x) { }
        void f1(int a) { cout << ++n1 << endl; }
};
int main(){
    derived d(10);
    d.f1(); //LINE-1
    return 0;
}
```

What will be the output/error?

a) `10`

b) `11`

c) `12`

d) `Compilation Error:  no matching function for call to derived::f1()`

Answer: d)
Explanation:
When we overload the base class function in the derived class, the base class function will not be available to call using the derived class object. So, it will be a compilation error at `LINE-1`.

Question 3

Consider the following code segment. *[MCQ, Marks 2]*

```
#include <iostream>
using namespace std;
class base {
public:
    void print() { cout << "C" << " "; }
};
class derived : public base {
public:
    void print() { cout << "C++" << " "; }
};
int main(){
    base *a1 = new base();
    base *b1 = new derived();
    a1->print();
    b1->print();
    return 0;
}
```

What will be the output?

a) `C C`

b) `C C++`

c) `C++ C`

d) `C++ C++`

Answer: a)
Explanation:
Since the function `base::print()` is non-virtual, the calling of `print()` are non-polymorphic. In compile time binding, the choice of the function implementation to be called depends on the type of the pointer. In our case, both pointers are having `base` class type. So, both pointer will call base class function `base::print()`.

Question 4

Consider the following code segment. *[MCQ, Marks 2]*

```
#include<iostream>
using namespace std;
class A{
    public:
        int a;
        A(int x) : a(x) { }
};
class B : protected A{
    int b;
    public:
        B(int x, int y) : b(y), A(x) { }
};
int main(){
    B t1(1,2);
    A t2(5);
    cout << t1.a; //LINE-1
    cout << t2.a; //LINE-2
    return 0;
}
```

Which line will generate compilation error in the `main()` function?

a) `LINE-1`

b) `LINE-2`

c) `Both LINE-1 and LINE-2`

d) `No compilation error`

Answer: a)
Explanation:
The data member `a` is declared as public in class `A`. So, we can access this from anywhere using the class `A` object. So, `LINE-2` is fine. But the same data member becomes protected when we inherit class `A` from class `B` as it is protected inheritance. So, `LINE-1` will give compilation error.

Question 5

Consider the following code segment. *[MCQ, Marks 2]*

```
#include<iostream>
using namespace std;
class A{
    public:
        A() { cout<<"A "; }
        ~A() { cout << "~A "; }
};
class B : public A {
    public:
        B() { cout << "B "; }
        ~B() { cout << "~B "; }
};
class C : public B{
    B b;
    public:
        C() { cout << "C "; }
        ~C() { cout << "~C "; }
};
int main(){
    C obj;
    return 0;
}
```

What will be the output?

a) `A B C ~C ~B ~A`

b) `A A B C ~C ~B ~A ~A`

c) `A B A B C ~C ~B ~A ~B ~A`

d) `A A B C ~C ~A ~A`

Answer: c)
Explanation:
When an object of `class C` is being instantiated, constructor of `class B` is called, which will again call constructor of `class A` that will print `A B` first. Then the data member of `class C` is created, which will again print `A` then print `B`. Then at last `C` is printed from the constructor of `class C`. After the end of `main()` function, reverse of already printed sequence will be printed from the destructor of the classes. So, the answer is c).

Question 6

Consider the following code segment. *[MSQ, Marks 2]*

```
#include<iostream>
using namespace std;
class base{
    public:
        static void func() { cout << "C++" << endl; }
};
class derived : private base {
    public:
        derived() { ____________________; } //LINE-1
};
int main(){
    derived t1;
    return 0;
}
```

Fill in the blank at `LINE-1` such that the output is `C++`.

a) `(new base)->func()`

b) `base::func()`

c) `base.func()`

d) `base::func`

Answer: a), b)
Explanation:
It can be observed that the `func()` function needs to be called from class `derived` in order to print `C++`. So, it can be called using class name or temporary object. So, option a) and b) are correct.

Question 7

Consider the following code segment. *[MSQ, Marks 2]*

```
#include <iostream>
using namespace std;
class A1{
    public:
        int t1;
};
class A2 : private A1{
    public:
        int t2;
        int sum(){ return t1 + t2; }
};
int main(){
    A1 b;
    A2 d;
    b.t1 = 10;          //LINE-1
    d.t1 = 20;          //LINE-2
    d.t2 = 30;          //LINE-3
    cout << d.sum();    //LINE-4
    return 0;
}
```

Which line/s in the `main` function will generate compilation error?

a) `LINE-1`

b) `LINE-2`

c) `LINE-3`

d) `LINE-4`

Answer: b)
Explanation:
In private inheritance, public members of base class appear as private members in derived class. Hence, in `LINE-2`, the code `d.t1 = 20;` gives error.
This question is intentionally made as MSQ

Question 8

Consider the following code segment. *[MSQ, Marks 2]*

```
#include<iostream>
using namespace std;
class B{
    int x;
    public:
        B(int _x) : x(_x){ }
        int fun(){ return x; }
};
class D : public B{
    int y;
    public:
        D(int _x, int _y) : __________________{} //LINE-1
        void fun(){ cout << B::fun() << y; }
};
int main(){
    D *b2 = new D(1,0);
    b2->fun();
    return 0;
}
```

Fill in the blank at `LINE-1` such that the program will print `10`.

a) `B(_x), y(_y)`

b) `B(_y), y(_x)`

c) `y(_y), B(_x)`

d) `y(_x), B(_y)`

Answer: a), c)
Explanation:
It is noted that data member of class `B` should be assigned with value 1 and data member of class `D` i.e. `y` should be assigned with value 0. Hence, option a) and c) are correct.
Note that, we need to call Base class constructor in order to assign value to its data member.

Question 9

Consider the following code segment. *[MCQ, Marks 2]*

```
#include<iostream>
using namespace std;
class myClass1{
    int x;
public:
    myClass1(int a) : x(a){}
    void fun(){
        cout << x;
    }
};
class myClass2 : public myClass1{
    int y;
public:
    myClass2(int a, int b) : myClass1(a), y(b) {}
    void fun(){
        cout << y;
    }
};
int main(){
    myClass2 m(1,2);
    ____________________; //LINE-1
    return 0;
}
```

Fill in the blank at `LINE-1` such that the program will print `1`.

a) `myClass1.m::fun()`

b) `m.myClass1::fun()`

c) `m.myClass1.fun()`

d) `myClass1.m.fun()`

Answer: b)
Explanation:
Since the function `fun()` needs to be called from the base class, the appropriate syntax for function call is `m.myClass1::fun()`.

Programming Questions

Question 1

Complete the program with the following instructions.

- Fill in the blank at `LINE-1` to complete the inheritance statement for class `Rectangle`,
- Fill in the blanks at `LINE-2` and `LINE-3` to complete the return statement.

The program must satisfy the given test cases. *Marks: 3*

```
#include<iostream>
using namespace std;
class Area{
    public:
        double getVal(int x, int y){ return (x*y); }
};
class Perimeter{
    public:
        double getVal(int x, int y){ return (2*(x+y)); }
};
class Rectangle : ____________________________________{ //LINE-1
    int x, y;
    public:
        Rectangle(int _x, int _y) : x(_x), y(_y){ }
        double getArea(){ return ___________________; } //LINE-2
        double getPerimeter(){ return ____________________; } //LINE-3
};
int main(){
    int a, b;
    cin >> a >> b;
    Rectangle r(a,b);
    cout << r.getArea() << ", " << r.getPerimeter();
    return 0;
}
```

Public 1

```
Input: 5 8
Output: 40, 26
```

Public 2

```
Input: 2 5
Output: 10, 14
```

Private 1

```
Input: 56 87
Output: 4872, 286
```

Answer:

```
LINE-1:  public Area, public Perimeter
LINE-2:  Area::getVal(x,y)
```

`LINE-3: Perimeter::getVal(x,y)`

Explanation:

The class `Rectangle` must be inherited from both `Area` and `Perimeter` classes. So, at `LINE-1`, we use

`class Rectangle : public Area, public Perimeter`

Note that any of public, protected, or private inheritance will work in this case, classes may be put in any order, and private inheritance may be implied by skipping the specifier/s for inheritance. Hence, there are several fill-ups that will work as long as both classes `Area` and `Perimeter` are listed.

The function `getVal()` is defined in both `Area` and `Perimeter` classes. To resolve the ambiguity, we need to use `Area::getVal(x,y)` at `LINE-2` to call `getVal()` from class `Area` and `Perimeter::getVal(x,y)` at `LINE-3` to call `getVal()` from class `Perimeter`.

Question 2

Consider the following program with the following instructions.

- Fill in the blanks at `LINE-1` to complete the inheritance statement
- Fill in the blank at `LINE-2` to complete the constructor statement

The program must satisfy the sample input and output. *Marks: 3*

```
#include<iostream>
using namespace std;
class B1{
protected:
    int b1;
    public:
        B1(int b) : b1(b){}
};
class B2{
    protected:
        int b2;
    public:
        B2(int b) : b2(b){}
};
class D : _________________________{ //LINE-1
    int d;
    public:
        D(int x) : ___________________________{} //LINE-2
        void show(){
            cout << d << ", " << b1 << ", " << b2;
        }
};
int main(){
    int x;
    cin >> x;
    D t1(x);
    t1.show();
    return 0;
}
```

Public 1

```
Input: 8
Output: 8, 10, 12
```

Public 2

```
Input: 5
Output: 5, 7, 9
```

Private

```
Input: 50
Output: 50, 52, 54
```

Answer:

`LINE-1:` `public B1, public B2`

OR

`LINE-1:` `protected B1, protected B2`

`LINE-2:` `B1(x+2), B2(x+4), d(x)` OR in any order of them

Explanation:

The function `show()` of class `D` is accessing protected member of both class `B1` and class `B2`. This can be done when `D` class is inherited from class `B1` and `B2`.

So, `LINE-1` will be filled as `public B1, public B2` or protected inheritance in any order.

As per the test cases, the constructor at `LINE-2` needs to be filled as `B1(x+2), B2(x+4), d(x)` or in any order.

Question 3

Consider the following program. Fill in the blanks as per the instructions given below:

- at `LINE-1, LINE-2`, and LINE-3 to complete the constructor initialization,
- at `LINE-4, LINE-5`, and `LINE-6` to complete return statements,

such that it will satisfy the given test cases. *Marks: 3*

```
#include<iostream>
using namespace std;
class Step{
    int a;
    public:
        Step(int _a = 0);
        int sum();
};
class Step1 : public Step{
    int b;
    public:
        Step1(int _a = 0, int _b = 0);
        int sum();
};
class Step2 : public Step1{
    int c;
    public:
        Step2(int _a = 0, int _b = 0, int _c = 0);
        int sum();
};
Step::Step(int _a) : ________________________ {} //LINE-1
Step1::Step1(int _a, int _b) : ____________________ {} //LINE-2
Step2::Step2(int _a, int _b, int _c) : ________________ {} //LINE-3
int Step::sum(){ return _____________; } //LINE-4
int Step1::sum(){ return _____________; } //LINE-5
int Step2::sum(){ return _____________; } //LINE-6
int main(){
    int a, b, c;
    cin >> a >> b >> c;
    Step aObj(a);
    Step1 bObj(a, b);
    Step2 cObj(a, b, c);
    cout << aObj.sum() << ", " << bObj.sum() << ", " <<     cObj.sum();
    return 0;
}
```

Public 1

```
Input: 5 6 7
Output: 5, 11, 18
```

Public 2

```
Input: 10 20 30
Output: 10, 30, 60
```

Private

```
Input: -5 0 5
Output: -5, -5, 0
```

Answer:

```
LINE-1:  a(_a)
LINE-2:  Step(_b), b(_a)
LINE-3:  Step1(_b,_c), c(_a)
LINE-4:  a
LINE-5:  Step::sum() + b
LINE-6:  Step1::sum() + c
```

Explanation:

The initialization lists at `LINE-1, LINE-2` and `LINE-3` are as follows:

```
LINE-1:  a(_a)
LINE-2:  Step(_b), b(_a)
LINE-3:  Step1(_b,_c), c(_a)
```

The return statement of the `sum()` functions at `LINE-4, LINE-5` and `LINE-6` are as follows:

```
LINE-4:  a
LINE-5:  Step::sum() + b
LINE-6:  Step1::sum() + c
```