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 16, 2023

Question 1

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

```
#include <iostream>
using namespace std;
class Interest {
    protected:
        double i;
    public:
        Interest(double _i) : i(_i) {}
        void calculate() { cout << i << endl; }
};
class FDInterest : public Interest {
    public:
        FDInterest(double _i) : Interest(_i) {}
        void calculate(double prin) { cout << i * prin << endl; }
};
int main(){
    FDInterest i1(6.75);
    i1.calculate();    //LINE-1
    return 0;
}
```

What will be the output/error?

a) `6.75`

b) `0`

c) `675`

d) `Compilation error: no matching function for call to 'FDInterest::calculate()'`

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

Question 2

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

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

What will be the output?

a) `C Programming`
   `C++ Programming`

b) `C++ Programming`
   `C Programming`

c) `C Programming`
   `C Programming`

d) `C++ Programming`
   `C++ Programming`

Answer: c)
Explanation:
Since `print()` is a non-virtual function, the binding of the function calls `a1->print()` and `b1->print()` depend on the type of the pointers. In our case, both pointers are having `C` class type. So, both pointer will call base class function `C::print(.)`.

Question 3

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

```
#include<iostream>
using namespace std;
class One{
    public:
        One() { cout<<"1 "; }
        ~One() { cout << "-1 "; }
};
class Two : public One {
    public:
        Two() { cout << "2 "; }
        ~Two() { cout << "-2 "; }
};
class Three : public One{
    Two b;
    public:
        Three() { cout << "3 "; }
        ~Three() { cout << "-3 "; }
};
int main(){
    Three t1;
    return 0;
}
```

What will be the output?

a) `1 2 3 -3 -2 -1`

b) `1 1 2 3 -3 -1 -1`

c) `1 3 -3 -1`

d) `1 1 2 3 -3 -2 -1 -1`

Answer: d)
Explanation:
When an object of class `Three` is being instantiated, the constructor of class `One` is called, which will print "1" first. Then data member of class `Three` is created, which will again print "1" then print "2". Then at last "3" is printed from constructor of class `Three`. After the end of `main()` function, reverse of already printed sequence will be printed from the destructor of the classes. So, the answer is (d).

Question 4

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

```
#include<iostream>
using namespace std;
class Base{
    public:
        void print() { cout << "Base" << endl; }
};
class Derived : protected Base {
    public:
        Derived() { _________________ } //LINE-1
};
int main(){
    Derived t1;
    return 0;
}
```

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

a) `Base::print();`

b) `Base.print();`

c) `(new Base)->print();`

d) `Base->print();`

Answer: a), c)
Explanation:
It can be seen that the `print()` function needs to be called from `class B` constructor in order to print `Base`. So, it can be called using class name or temporary object. So, option a) and c) are correct.

Question 5

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

```
#include <iostream>
using namespace std;
class B {
    protected:
        int X;
    public:
        B(int i = 0) : X(i) {}
};
class D : public B {
    B b;
    public:
        D(B b1, int i = 0) : B(i), b(b1) {}
        void print1() { cout << X << endl; }   // LINE-1
        void print2() { cout << b.X << endl; } // LINE-2
};
int main() {
    B b(5);
    D d(b, 10);
    d.print1();
    d.print2();
    return 0;
}
```

What will be the output/error?

a) `5 10`

b) `0 10`

c) `Compilation error generated from LINE-1`

d) `Compilation error generated from LINE-2`

Answer: d)
Explanation:
Both the `print1()` and `print2()` functions are trying to access protected data member `X` of `class B`. `class D` has two data members. One is b of type `B` which is private and the other is `X` which is inherited from `B` class, and due to inheritance property, it is protected. So in `D` class, variable `X` is easily accessible from public function `print1()` in `LINE-1`. But in `LINE-2`, `print2()` function is trying to access data member of `object B`, which is protected in `B` class. So it can't be accessed from outside `B` class.

Question 6

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

```
#include <iostream>
using namespace std;
class A{
    public:
        A(int i){ cout << "A::" << i << " "; }
        ~A(){ cout << "~A "; }
};
class B : public A{
    public:
        B(int i) : A(i){ cout << "B::" << i << " "; }
        ~B(){ cout << "~B "; }
};
class C : public B{
    public:
        C(int i) : B(i){ cout << "C::" << i << " "; }
        ~C(){ cout << "~C "; }
};
C *dp;
void caller(){
    dp = new C(1); //LINE-1
}
int main(){
    C d(2); //LINE-2
    {
        C d(3); //LINE-3
    }
    caller(); //LINE-4
    delete dp; //LINE-5
    return 0;
}
```

What will be the output?

a) `A::2 B::2 C::2 A::3 B::3 C::3 A::1 B::1 C::1 ~C ~B ~A ~C ~B ~A ~C ~B ~A`

b) `A::2 B::2 C::2 A::3 B::3 C::3 ~C ~B ~A A::1 B::1 C::1 ~C ~B ~A ~C ~B ~A`

c) `C::2 B::2 A::2 C::3 B::3 A::3 C::1 B::1 A::1 ~C ~B ~A ~A ~B ~A ~C ~B ~A`

d) `C::2 B::2 A::2 C::3 B::3 A::3 C::1 B::1 A::1 ~A ~B ~C ~A ~B ~C ~A ~B ~C`

Answer: b)
Explanation:
The constructors are invoked in top-down order in class hierarchy, whereas destructors are invoked bottom-up order in class hierarchy.
At `LINE-2`, as the object of class C is instantiated. Hence, the output will be `A::2 B::2 C::2`.
Another object of class C is instantiated at `LINE-3`, hence output is `A::3 B::3 C::3`. However, the object gets deleted as soon the scope ends, hence it generates output `~C ~B ~A`.
Next, at `LINE-4` as the function `caller()` is called. The code at `LINE-1` generates another object of class C that generates the output `A::1 B::1 C::1`. The same object gets deleted at `LINE-5`. Hence, the output is `~C ~B ~A`.

Finally, the object created at `LINE-2` gets deleted at the end of the program, that again generates output as ∼`C` ∼`B` ∼`A`.

Question 7

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

```
#include<iostream>
using namespace std;
class Base {
    public:
        void f() { cout<< "Base::f()"; }
};
class Derived : public Base {
    public:
        void f() { cout<<"Derived::f()"; };
};
main() {
    Derived obj;
    ________________; //LINE-1
    return 0;
}
```

Fill in the blank at `LINE-1` so that the program will print `Base::f()`.

a) `Base.obj.f()`

b) `Base.obj::f()`

c) `obj.Base::f()`

d) `Base::obj.f()`

Answer: c)
Explanation:
As the function `f()` need to be called from the base class `Base`, the appropriate syntax for function call is `obj.Base::f()`.
This question is intentionally made as MSQ

Question 8

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

```
#include<iostream>
using namespace std;
class Staff{
    string name;
    public:
        Staff(string _name = "unknown") : name(_name){}
        void print1(){ cout << name << " "; }
};
class Teacher : protected Staff{
    string deptName;
    public:
        Teacher(string _name, string _deptName) : Staff(_name), deptName(_deptName){}
        void print2(){ cout << deptName << " "; }
};
int main(){
    Teacher t("Partha", "CSE");
    t.print1();     //LINE-1
    t.print2();     //LINE-2
    return 0;
}
```

What will be the output/error?

a) `Partha CSE`

b) `unknown CSE`

c) `Compilation error at LINE-1: void 'Staff::print1()' is inaccessible in this context`

d) `Compilation error at LINE-2: void 'Staff::print2()' is inaccessible in this context`

Answer: c)
Explanation:
Due to protected inheritance, the function `print1()` becomes protected in class `Teacher`. Hence, `t.print1()` generates a compiler error.

Question 9

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

```
#include <iostream>
using namespace std;
class A1 {
    protected:
        int t1;
    public:
        A1(int _t1) : t1(_t1) { }
};
class A2 : public A1 {
    protected:
        int t2;
    public:
        A2(int _t1, int _t2) : A1(_t1), t2(_t2) { }
};
class A3 : private A2 {
    public:
        A3(int _t1, int _t2) : ____________ { }    //LINE-1
        void print() { cout << t1 << " " << t2; }
};
int main() {
    A3 d(10, 20);
    d.print();
    return 0;
}
```

Fill in the blank at `LINE-1` such that the program will print `10 20`.

a) `A2(_t1, _t2)`

b) `A2(_t2, _t1)`

c) `A1(_t1), A2(_t2)`

d) `A2(_t1), A2(_t1, _t2)`

Answer: a)
Explanation:
The output suggests that data member of class `A1 and A2` will be assigned with 10 and 20 respectively. This can be done in `LINE-1` as `A2(_t1, _t2)`.

Programming Questions

Question 1

Complete the program with the following instructions.

- Fill in the blank at `LINE-1` with appropriate initializer list,
- Fill in the blanks at `LINE-2` and `LINE-3` to complete the return statements.

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

```
#include <iostream>
using namespace std;
class Radius{
    public:
        int radius;
        Radius(int r) : radius(r) { }
};
class Volume : public Radius{
    public:
        Volume(int _r) : Radius(_r){}
        double getVal(){ return 1.33 * 3.14 * radius * radius * radius; }
};
class Area : public Radius{
    public:
        Area(int _r) : Radius(_r){}
        double getVal(){ return 4 * 3.14 * radius * radius; }
};
class Sphere : public Volume, public Area{
    public:
        Sphere(int _r) : _______________________{}    //LINE-1
        double getVolume(){ return __________________; } //LINE-2
        double getArea(){ return __________________; } //LINE-3
};
int main(){
    int a;
    cin >> a;
    Sphere s(a);
    cout << s.getArea() << ", " << s.getVolume();
    return 0;
}
```

Public 1

```
Input: 3
Output: 113.04, 112.757
```

Public 2

```
Input: 10
Output: 1256, 4176.2
```

Private 1

```
Input:5
Output: 314, 522.025
```

Answer:

```
LINE-1:  Volume(_r), Area(_r)
LINE-2:  Volume::getVal()
LINE-3:  Area::getVal()
```

Explanation:

The class `Sphere` is inherited from both `Area` and `Volume` classes. So, at `LINE-1`, we use `class Rectangle :  Volume(_r), Area(_r)` for the constructor definition.

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

Question 2

Consider the following program with the following instructions.

- Fill in the blank at `LINE-1` with appropriate keyword.
- Fill in the blanks at `LINE-2 and LINE-3` to complete the constructor statements.

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

```
#include <iostream>
using namespace std;
class Vehicle{
    string vehicleName;
    int noOfWheels;
    protected:
        Vehicle(string s, int w) : vehicleName(s), noOfWheels(w) { }
    public:
        ______ void vehicleDetails(const Vehicle&); //LINE-1
};
class Twowheeler : public Vehicle{
    public:
        Twowheeler(string n) : _______________ { } //LINE-2
};
class Fourwheeler : public Vehicle{
    public:
        Fourwheeler(string n) : _________________ { } //Line-3
};
void vehicleDetails(const Vehicle &v){
    cout << v.vehicleName << ": ";
    if(v.noOfWheels == 2)
        cout << "Two Wheeler";
    else
        cout << "Four Wheeler";
}
int main(){
    string s;
    int n;
    Vehicle *v;
    cin >> s >> n;
    if(n==2)
        v = new Twowheeler(s);
    else
        v = new Fourwheeler(s);
    vehicleDetails(*v);
    return 0;
}
```

Public 1

```
Input: Car 4
Output: Car: Four Wheeler
```

Public 2

```
Input: Cycle 2
Output: Cycle: Two Wheeler
```

Private

```
Input: Bus 4
Output: Bus: Four Wheeler
```

Answer:
```
LINE-1:  friend
LINE-2:  Vehicle(n,2)
LINE-3:  Vehicle(n,4)
```
Explanation:
The global function `vehicleDetails()` needs to access private members of class `Vehicle`. So, it should be a `friend` function of class Vehicle. `LINE-1` will be filled with `friend`.
From `LINE-2`, constructor of class `Vehicle` needs to be called with `noOfWheels` value as 2. It can be done as `Vehicle(n, 2)`.
In the similar way, `LINE-3` will be filled as `Vehicle(n, 4)`.

Question 3

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

- Fill in the blank at `LINE-1` to complete the inheritance statement.
- Fill in the blank at `LINE-2` with the appropriate initializer list,

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

```
#include<iostream>
using namespace std;
class Test1{
    protected:
        int t1;
    public:
        Test1(int b) : t1(b){}
};
class Test2{
    protected:
        int t2;
    public:
        Test2(int b) : t2(b){}
};
class ReTest : _________________________________{ //LINE-1
    int t;
    public:
        ReTest(int x) : ________________________________{} //LINE-2
        void show(){
            cout << t << ", " << t1 << ", " << t2;
        }
};
int main(){
    int x;
    cin >> x;
    ReTest t1(x);
    t1.show();
    return 0;
}
```

Public 1

```
Input: 3
Output: 3, 8, 13
```

Public 2

```
Input: 10
Output: 10, 15, 20
```

Private

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

Answer:
`LINE-1:  public Test1, public Test2` or protected inheritance
`LINE-2:  Test1(x+5), Test2(x+10), t(x)` or in any order
Explanation:
The function `show()` of class `ReTest` is accessing protected member of both class `Test1` and class `Test2`. This can be done when `ReTest` class is inherited from class `Test1` and `Test2`.
So, `LINE-1` will be filled as
`public Test1, public Test2 or protected inheritance`
As per the test cases, the constructor at `LINE-2` needs to be filled as
`Test1(x+5), Test2(x+10), t(x) or in any order`