Programming in Modern C++: Assignment Week 6

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 24, 2023

Question 1

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

```
#include<iostream>
using namespace std;
class Base{
    public:
        void fun1() { cout << "1" ; }
        virtual void fun2() { cout << "3" ; }
};
class Derived : public Base{
    public:
        virtual void fun1() { cout << "2" ; }
        void fun2() { cout << "4" ; }
};
int main(){
    Base *t = new Derived();
    t->fun1();
    t->fun2();
    return 0;
}
```

What will be the output?

a) `13`

b) `14`

c) `23`

d) `24`

Answer: b)
Explanation:
As `fun1()` is a non-virtual function at the base class, for the `t->fun1()` function call static binding is done. So, the function of pointer type will be called.
As `fun2()` is a virtual function, for the `t->fun2()` function call dynamic binding is done. So, the function of object type will be called.

Question 2

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

```
#include<iostream>
using namespace std;
int x = 0;
class ClassA{
    public:
        ClassA(){ x = x+2; }
        ~ClassA() { x = x-1; }
};
class ClassB : public ClassA{
    public:
        ClassB(){ x = x+3; }
        ~ClassB(){ x = x-2; }
};
void fun(){
    ClassB t;
    ClassA *t1 = new ClassB();
    cout << x << " ";
    delete t1;
}
int main(){
    fun();
    cout << x;
    return 0;
}
```

What will be the output/error?

a) `10 6`

b) `10 4`

c) `8 6`

d) `8 4`

Answer: a)
Explanation:
When the function `fun` is called, an object of class `ClassB` is created, which increase the value of the global variable by $(3 + 2) = 5$. Again an object of class `ClassB` is created with `new` which will increase global variable `x` by 5. So, 10 is printed first. When it is returned from the function, destructor for both object would be called. In this process, for the object `(t1)`, only base class destructor is called because of non-virtual-ness. But, for the object `(t)`, both class destructor would be called. So, `x` is decreased by only 4. So, 6 will be printed.

Question 3

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 "; }
        virtual ~B() { cout<<"~B "; }
};
class C : public B{
    public:
        C() { cout<<"C "; }
        ~C() { cout<<"~C "; }
};
int main(){
    A *t1 = new C;
    delete t1;
    return 0;
}
```

What will be the output?

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

b) `A B C ~C ~B`

c) `A B C ~B ~A`

d) `A B C ~A`

Answer: d)
Explanation:
When the object of class `C` is created, it calls constructor of class `C` which in turn calls constructor of class `B` and `A` respectively. So, it will print `A B C`.
Whenever, the object is deleted, it calls destructor of class `A` first. The destructor of class `A` is not virtual, so it will not call child class destructor. So, final result will be `A B C ~A`.

Question 4

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

```
#include <iostream>
using namespace std;
class Virtual {
    public:
        virtual void fun() = 0;   //LINE-1
};
void Virtual::fun() {
    cout << "Pure virtual function";
}
int main() {
    Virtual m; // LINE-2
    Virtual *p = new Virtual(); // LINE-3
    p->fun(); // LINE-4
    return 0;
}
```

Which line/s will give you error?

a) `LINE-1`

b) `LINE-2`

c) `LINE-3`

d) `LINE-4`

Answer: b), c)
Explanation:
Any abstract base class cannot be instantiated and hence will give an error at `LINE-2`. Also, we cannot create abstract class object using new operator and hence will give an error at `LINE-3`.

Question 5

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

```
#include<iostream>
using namespace std;
class Base{
    public:
        virtual void fun() { }
};
class Derived : public Base{
    public:
        void fun(double i) { }
};
int main(){
    Derived t1;
    Base *t2 = new Derived();
    t1.fun();          //LINE-1
    t1.fun(3.14);      //LINE-2
    t2->fun();         //LINE-3
    t2->fun(3.14);     //LINE-4
    return 0;
}
```

Which line/s will give you error?

a) `LINE-1`

b) `LINE-2`

c) `LINE-3`

d) `LINE-4`

Answer: a), d)
Explanation:
The function `fun()` of class `Base` is overloaded in class `Derived`. So, base class function become hidden for derived class. So, `LINE-1` will give error. On the other hand, class `Base` doesn't have `fun(double)` in its definition. So, `LINE-4` will give an error.

Question 6

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

```
#include<iostream>
using namespace std;
class classA{
    public:
        virtual void f(){ cout << "A::f() "; }
        void g(){ cout << "A::g() "; }
        void h(){ cout << "A::h() "; }
};
class classB : public classA{
    public:
        void f(){ cout << "B::f() "; }
        virtual void g(){ cout << "B::g() "; }
        void h(){ cout << "B::h() "; }
};
class classC : public classB{
    public:
        void f(){ cout << "C::f() "; }
        void g(){ cout << "C::g() "; }
        virtual void h(){ cout << "C::h() "; }
};
int main(){
    classC cb;
    classB &bb = cb;
    bb.f();
    bb.g();
    bb.h();
    return 0;
}
```

What will be the output?

a) `A::f() B::g() C::h()`

b) `C::f() C::g() B::h()`

c) `C::f() B::g() B::h()`

d) `C::f() C::g() C::h()`

Answer: b)
Explanation:
In class `classB`, the functions `f()` and `g()` are virtual functions. As `bb` refers to the object `cb`, the output will be `C::f() C::g() B::h()`.

Question 7

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

```
#include<iostream>
using namespace std;
class A{
    public:
        virtual void fun(){ cout << "1 "; }
};
class B : public A{
    public:
        void fun(){ cout << "2 "; }
};
class C : public B{
    public:
        void fun(){ cout << "3 "; }
};
int main(){
    C *cb = new C;
    ________________; //LINE-1
    return 0;
}
```

Fill in the blank at `LINE-1` so that the program will print `2`.

a) `cb->B::fun()`

b) `B::fun()`

c) `B::cb->fun()`

d) `cb->fun()`

Answer: a)
Explanation:
As `cb` is a pointer to an object of `class C`, we can call `fun()` from `class B` as `cb->B::fun()` such that it will print `2`.

Question 8

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

```
#include<iostream>
using namespace std;
class Vehicle{
    public:
        virtual void run() = 0;
        virtual void stop() = 0;
};
class Car : public Vehicle{
};
class MotorCycle : public Vehicle{
    public:
        void run(){}
        void stop(){}
};
class Truck : public Car{
    public:
        void run(){}
        void stop(){}
};
class SportsCar : public Car{
    public:
        void run(){}
        virtual void nitro() = 0;
        void stop(){}
};
void SportsCar::nitro(){}
class SUV : public Car{
    public:
        void run(){}
};
```

Identify the abstract classes.

a) `Vehicle, Car, MotorCycle`

b) `Vehicle, Car, SUV`

c) `Vehicle, Car`

d) `Vehicle, Car, SportsCar, SUV`

Answer: d)
Explanation:
A class having at least one pure virtual function is an abstract class. At the same time, a pure virtual function remains pure virtual until it is overridden by the derived class. Hence, the correct option is d).

Question 9

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

```
#include<iostream>
using namespace std;
class B{
    int b;
public:
    B(int i) : b(i) {}
    virtual void f(B *t) { cout << t->b << endl; }
};
class D : public B{
    int d;
public:
    D(int i=0, int j=0) : B(i), d(j) { }
    void f(D *t) { cout << t->d << endl; }
};
int main(){
    B *t1 = new D(1,2);
    t1->f(new D); //Line-1
    return 0;
}
```

What will be the output?

a) `0`

b) `1`

c) `2`

d) `Garbage`

Answer: a)
Explanation:
The function call at LINE-1 invokes derived class function with a temporary object as parameter. The temporary object of class D has data members with value 0 as constructor initializes both the data members with default value 0. Hence, the program will print 0.

Programming Questions

Question 1

Complete the program with the following instructions.

- Fill in the blank at `LINE-1` with proper access specifier.
- Fill in the blanks at `LINE-2` to declare `area()` as a pure virtual function.

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

```
#include<iostream>
using namespace std;
class Shape{
    ____________:     //LINE-1
        double ar;
    public:
        ______________________________;     //LINE-2
        void show(){
            cout << ar << " ";
        }
};
class Triangle : public Shape{
    int h, w;
    public:
        Triangle(int a, int b) : h(a), w(b){}
        void area(){
            ar = 0.5*h*w;
        }
};
class Circle : public Shape{
    int r;
    public:
        Circle(int a) : r(a){}
        void area(){
            ar = 3.14*r*r;
        }
};
int main(){
    int w,h,r;
    cin >> w >> h >> r;
    Shape *s1 = new Triangle(h,w);
    Shape *s2 = new Circle(r);
    s1->area();
    s2->area();
    s1->show();
    s2->show();
    return 0;
}
```

Public 1

```
Input: 1 2 3
Output: 1 28.26
```

Public 2

```
Input: 2 5 10
Output: 5 314
```

Private 1

```
Input: 3 5 7
Output: 7.5 153.86
```

Answer:

```
LINE-1:  protected OR public
LINE-2:  virtual void area() = 0
```

Explanation:

The data member of class `Shape` is being accessed from its child classes. Hence, the access specifier of the data member should be `protected` or `public` at `LINE-1`. the function area needs to be declared as pure virtual at `LINE-2` which can be done as `virtual void area() = 0;`

Question 2

Consider the following program with the following instructions.

- Fill in the blank at `LINE-1` with appropriate destructor declaration.
- Fill in the blank at `LINE-2` with appropriate initialization list of derived class constructor.

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

```
#include<iostream>
using namespace std;
class Base{
    public:
        Base(){ cout << "1 "; }
        Base(int n){ cout << n << " "; }
        ______________________;         //LINE-1
};
Base::~Base(){ cout << "2 "; }
class Derived : public Base{
    public:
        Derived(){ cout << "3 "; }
        Derived(int n) : ____________ { cout << n * 3 << " "; } //LINE-2
        virtual ~Derived(){ cout << "4 "; }
};
int main(){
    int i;
    cin >> i;
    Base *pt = new Derived(i);
    delete pt;
    return 0;
}
```

Public 1

```
Input: 5
Output: 5 15 4 2
```

Public 2

```
Input: 7
Output: 7 21 4 2
```

Private

```
Input: 50
Output: 50 150 4 2
```

Answer:
`LINE-1:` `virtual ~Base()`
`LINE-2:` `Base(n)`
Explanation:
At `LINE-1`, the destructor needs to be defined as virtual destructor, so that if the derived class object gets deleted it will be called automatically. Hence, `LINE-1` has to be filled with `virtual ~Base();`
The initialization-list required at `LINE-2` is `Base(n)`.

Question 3

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

- Fill in the blanks at `LINE-1` with an appropriate keyword.
- Fill in the blank at `LINE-2` with an appropriate function declaration.
- Fill in the blank at `LINE-3` with an appropriate statement.

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

```
#include<iostream>
using namespace std;
class B{
    protected:
        int s;
    public:
        B(int i=0) : s(i){}
        _______ ~B(){ }           //LINE-1
        ________________________; //LINE-2
};
class D : public B{
    public:
        D(int i) : B(i) {}
        ~D();
        int fun(int x){
            return s+x;
        }
};
class Z{
    public:
        void fun(int a, int b){
            B *t = _________;     //LINE-3
            int i = t->fun(b);
            cout << i << " ";
            delete t;
        }
};
D::~D(){ cout << --s << " "; }
int main(){
    int i, j;
    cin >> i >> j;
    Z w;
    w.fun(i,j);
    return 0;
}
```

Public 1

```
Input: 5 3
Output: 8 4
```

Public 2

```
Input: 2 8
Output: 10 1
```

Private

```
Input: 7 6
Output: 13 6
```

Answer:

```
LINE-1:  virtual
LINE-2:  virtual int fun(int) = 0
LINE-3:  new D(a)
```

Explanation:

The destructor of `B` class needs to be declared as virtual in order to call `D` class destructor at the time of deletion. The function `fun()` can be declared as a pure virtual function at `LINE-2` as `virtual int fun(int) = 0;`. We can't instantiate abstract class. So, `LINE-3` can be filled as `new D(a)`.