Programming in C++: Assignment Week 5

Total Marks : 20

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

April 3, 2017

Question 1

Look at the code snippet bellow. Find out the sequence in which the function `area()` associated with `Rectangle & Triangle class` will be called. *Mark 1*

```
class Polygon {
   protected:
      int width, height;
   public:
      void set_values (int a, int b)
          { width=a; height=b;}
 };

class Rectangle: public Polygon {
   public:
      int area ()
          { /* Function definition */ } // Area-1
};

class Triangle: public Polygon {
   public:
     int area ()
          { /* Function definition */ } // Area-2
};

int main () {
     Rectangle rect;
     Triangle trgl;
     rect.set_values (6,5);
     trgl.set_values (6,5);
     rect.area() ;
     trgl.area() ;
     return 0;
}
```

a) Area-2, Area-1

b) Area-1, Area-1

c) Area-1, Area-2

d) Area-2, Area-2

Answer: c)
Solution: By the order of calling functions

Question 2

Which of the following function will be invoked by `d.func(1)`? *Mark 1*

```
#include <iostream>
using namespace std;

class Base { public:
    int var_;
    void func(int){}
};

class Derived: public Base { public:
    int varD_;
    void func(int){}
};

int main() {
    Derived d;
    d.func(1);
    return 0;
}
```

a) `Base::func(int)`

b) `Derived::func(int)`

c) Compilation Error

d) `Base::func(int)` then `Derived::func(int)`

Answer: b) **Solution**: d is a Derived class object

Question 3

Find out the out put of the following Program.

```
#include <iostream>
using namespace std;

class Animal {
public:
    int legs = 4;
};
```

```
class Dog : public Animal {
public:
    int tail = 1;
};

int main() {
    Dog d;
    cout << d.legs;
    cout << d.tail;
    return 0;
}
```

a) 1 4

b) Compilation Error: Can't access Dog::tail

c) Compilation Error: Can't access Animal::legs

d) 4 1

Answer: d) **Solution**: legs is inherited from the Animal class

Question 4

Look at the code snippet bellow. Find out, which of the `show()` function will be called by calling `b->show()`. *Mark 1*

```
class Base {
public:
    void show() { }
};
class Derived :public Base {
public:
    void show() { }
};

int main() {
    Base* b;       //Base class pointer
    Derived d;     //Derived class object
    b = &d;
    b->show();
    return 0;
}
```

a) `show()` of Base class only

b) `show()` of Derived Class only

c) Both but, `show()` of Derived Class first then Base class

d) Both but, `show()` of Base class first then Derived Class

Answer: a)
Solution: Early Binding Occurs

Question 5

Look at the code snippet bellow. Find out, which of the `show()` function will be called by calling `b->show()` ? *Mark 1*

```
class Base {
public:
    virtual void show() { cout << "Base class"; }
};
class Derived :public Base {
public:
    void show() { cout << "Derived Class"; }
};

int main() {
    Base* b;       //Base class pointer
    Derived d;     //Derived class object
    b = &d;
    b->show();
    return 0;
}
```

a) `show()` of Base class only

b) `show()` of Derived Class only

c) Both but, `show()` of Derived Class first then Base class

d) Both but, `show()` of Base class first then Derived Class

Answer: b)
Solution: Late Binding Occurs as show() is virtual

Question 6

What will be the output of the following Code snippet? *Mark 1*

```
class B {
public:
    B() { cout << "B "; }
    ~B() { cout << "~B "; }
};
class C : public B {
public:
    C() { cout << "C "; }
    ~C() { cout << "~C "; }
};
class D : private C {
    B data_;
public:
    D() { cout << "D " << endl; }
    ~D() { cout << "~D "; }
};
int main() {
```

```
    {D d; }
    return 0;
}
```

a) B C B C D ~D ~C ~B ~C ~B

b) B C B D ~D ~B ~C ~B

c) B D B C ~C ~B ~D ~B

d) B C B C ~C ~B ~C ~B

Answer: b) **Solution**: In Inheritance, destructors are executed in reverse order of constructor execution

Question 7

Identify the correct statements? *Mark 1*

```
class base {
    public:
        int x;
    protected:
        int y;
    private:
        int z;
};

class publicDerived: public base
{
    //1. x is public
    //2. y is protected
    //3. z is accessible from publicDerived
};

class protectedDerived: protected base
{
    //4. x is public
    //5. y is protected
    //6. z is not accessible from protectedDerived
};

class privateDerived: private base
{
    //7. x is private
    //8. y is protected
    //9. z is not accessible from privateDerived
}
```

a) 1, 3 & 5

b) 2, 4 & 6

c) 6, 8 & 9

d) 2, 6 & 7

Answer: d)
Solution: By definition of public, private and protected inheritance

Question 8

Which lines of the following program will not compile? *Mark 1*

```
#include <iostream>                  // ---1
using namespace std;                 // ---2

class Base { protected:               // ---3
    int var_;                        // ---4
    public:                          // ---5
    Base():var_(0){}                 // ---6
};                                   // ---7

class Derived: public Base { public: // ---8
   int varD_;                        // ---9
   void print () { cout << var_; }   // ---10
};                                   // ---11

int main() {                         // ---12
   Derived d;                        // ---13

   d.var_ = 1;                       // ---14
   d.varD_ = 1;                      // ---15

   cout << d.var_ <<" " << d.varD_;  // ---16

   return 0;                         // ---17
}                                    // ---18
```

a) 6, 10, 14, 15

b) 6, 15

c) 4, 14, 16

d) 6, 14, 16

Answer: c)
Solution: Can't access protected data member

Question 9

Consider the following code snippet. Which of the following statement is true, when `t.getX()` is called ? *Mark 1*

```
class Test {
int x;
public:
    Test(int a):x(a){}
```

```
    virtual void show() = 0;
    int getX() { /* Function definition */ }
};

int main(void){
    Test t(5);
    t.getX();
    return 0;
}
```

a) Return an integer value

b) Compiler Error: cannot declare variable 't' to be of abstract

c) Compiler Error: cannot access private variable x

d) b & c

Answer: b)
Solution: Can't create object of a abstract class

Question 10

What will be the output of the following program? *Marks 1*

```
#include<iostream>
using namespace std;

class Shape {
public:
    int x, y;

    Shape(int a = 0, int b = 0): x(a), y(b) {}
    void draw()
    { cout << x << " " << y << " "; }
};
class Rectangle : public Shape {
public:
    int w, h;

    Rectangle(int a = 5, int b = 6): w(a), h(b), Shape(7, 8) {}

    void draw()
    { Shape::draw(); cout << w << " " << h ; }
};

int main() {
    Rectangle *r = new Rectangle(1,2);

    r-> draw();

    return 0;
}
```

a) `0 0 1 2`

b) `7 8 1 2`

c) `7 8 5 6`

d) `0 0 5 6`

Answer: b)
Solution: Shape(7, 8) initialize x and y as 7 and 8. Rectangle(1,2) initialize w and h as 1 and 2.

Programming Questions

Question 1

Consider the skeletal code given below and Fill the blank or remove the blank in the header of the functions to match the output for the given input. *Marks 2*

```
#include <iostream>
using namespace std;

class Shape {
protected:
    float l;
public:
    void getData(){ cin >> l; }

    ________ float calculateArea(){


        return l*l;
    }
};

class Circle : public Shape {
public:
    _________ float calculateArea(){
        return 3.14*l*l;
    }
};

int main()
{
    Shape * b;       //Base class pointer
    Circle d;     //Derived class object
    b = &d;
    b->getData();
    cout << b->calculateArea();

    return 0;
}
```

a. Input: 5
   Output: 78.5

b. Input: 28
   Output: 2461.76

c. Input: 45
   Output: 6358.5

Answer: virtual
Solution: Derived class function is called using a base class pointer. virtual function is resolved late, at runtime.

Question 2

Consider the skeletal code given below.Fill up the blank by following the instructions associated with each blank and complete the code, So that the output of the test cases would match *Marks: 3*

```
#include <iostream>
using namespace std;

class Area {
public:
    int calc(int l, int b) { return l*b; }
};

class Perimeter {
public:
    int calc(int l, int b) { return 2 * (l + b); }
};

/* Rectangle class is derived from classes Area and Perimeter. */
class Rectangle: _________________ { // Inherit the required base classes
private:
    int length, breadth;
public:
    Rectangle(int l, int b) : length(l), breadth(b) {}

    int area_calc() {
        /* Calls calc() of class Area and returns it. */

        ----------------
    }

    int peri_calc() {
        /* Calls calc() function of class Perimeter and returns it. */

        ----------------
    }
};

int main() {
```

```
    int l, b;
    cin >> l >> b;

    Rectangle r(l, b);

    cout << r.area_calc() << endl;
    cout << r.peri_calc() ;

    return 0;
}
```

Public-1

1. Input:
   4
   6
   Output:
   24
   20

2. Input:
   2
   3
   Output:
   6
   10

3. Input:
   65
   34
   Output:
   2210
   198

Answer

```
public Area, public Perimeter

return Area::calc(length, breadth);

return Perimeter::calc(length, breadth);
```

Solution: Multiple Inheritance

Question 3

Consider the skeletal code given below. *Marks: 3*
Fill up the blanks by following the instructions associated with it and complete the code, so that the output of the test cases should match. *Do not change any other part of the code.*

```
#include<iostream>
using namespace std;

class Polygon {
protected: int width, height;
public:
    Polygon(int a, int b) : width(a), height(b) {}

    // Declare the area function here
    --------------------------------

    void printarea() { cout << this->area() << ":"; }
};

class Rectangle : public Polygon {
```

```
public:
    Rectangle(int a, int b) : Polygon(a, b) {}
    int area() { return width*height; }
};

class Triangle : public Polygon {
public:
    Triangle(int a, int b) : Polygon(a, b) {}
    int area() { return width*height / 2; }
};

int main() {
    int h, w;
    cin >> h >> w;

    // Declare ppoly1 and ppoly2 as "pointer to Polygon" and dynamically allocate
    // a Rectangle and a Triangle objects respectively held by these pointers
    ________________________________ // Rectangle object of h and w
    ________________________________ // Triangle object of h and w

    ppoly1->printarea(); // For Rectangle object
    ppoly2->printarea(); // For Triangle object

    delete ppoly1;
    delete ppoly2;

    return 0;
}
```

Public 1

```
Input:
4
5
Output: 20:10:
```

Public 2

```
Input:
30
15
Output: 450:225:
```

Private

```
Input:
8
6
Output: 48:24:
```

Answer

```
virtual int area(void) = 0; or virtual int area(void) { return 0; }

Polygon * ppoly1 = new Rectangle(h, w);
Polygon * ppoly2 = new Triangle(h, w);
```

Solution: Late binding occurs

Question 4

Fill the blank by defining the proper constructor where name of the parameter should be `nc`
Marks 2

```
#include <iostream>

using namespace std;

class Engine {
public:
    Engine(int nc) :cylinder(nc){}

    void start() {
        cout << getCylinder() << " cylinder engine started" ;
    };

    int getCylinder() {
        return cylinder;
    }

private:
    int cylinder;
};
```

```
class Car : private Engine
{    // Car has-a Engine
public:
//Define the constructor to run the code. Name of the parameter should be 'nc'
 ------------------------
    void start() {
        cout << "car with " << Engine::getCylinder() <<
            " cylinder engine started" << endl;

        _________________________// Call the start function of Engine
    }
};

int main()
{
    int cylin;
    cin >> cylin;
    Car c(cylin);
    c.start();
    return 0;
}
```

a. Input: 8
   Output: car with 8 cylinder engine started
   8 cylinder engine started

b. Input:10
   Output: car with 10 cylinder engine started
   10 cylinder engine started

c. Input:4
   Output: car with 4 cylinder engine started
   4 cylinder engine started

Answer

```
Car(int nc) : Engine(nc) { }
Engine::start();
```

Solution: Use of Private Inheritance