Programming in C++: Assignment Week 5

Total Marks : 20

August 19, 2017

Question 1

What will be the order of construction and destruction of the following code segment for the objects instantiation `D d1(1, 2); E e(4);` *Mark 4*

```
class B { protected: int data_;
    public:
        B(int d = 0) : data_(d) { cout << "B::B(int): " << data_ << endl; }
       ~B() { cout << "B::~B(): " << data_ << endl; }
};

class D: public B {protected: int info_;
    public:
        D(int d, int i) : B(d), info_(i) {
            cout << "D::D(int, int): " << data_ << ", " << info_ << endl; }
        D(int i) : info_(i) {
            cout << "D::D(int): " << data_ << ", " << info_ << endl; }
        ~D() { cout << "D::~D(): " << data_ << ", " << info_ << endl; }
};

class E: protected B { int msg_;
    public:
        E(int i, int m): B(i), msg_(m) {
            cout << "E::E(int, int): " << data_ << ", " << msg_ << endl; }
        E(int m) : msg_(m) {
            cout << "E::E(int, int):" << data_ << ", " << msg_ << endl; }
        ~E() {cout << "E::~E():" << data_ << ", " << msg_ << endl; }
};
```

a) D::D(int, int): 1, 2
B::B(int): 1
E::E(int, int):0, 4
B::B(int): 0
E::~E():0, 4
B::~B(): 0
D::~D(): 1, 2
B::~B(): 1

b) B::B(int): 1
D::D(int, int): 1, 2
B::B(int): 0
E::E(int, int):0, 4
E::~E():0, 4
B::~B(): 0
D::~D(): 1, 2
B::~B(): 1

c) B::B(int): 1
D::D(int, int): 1, 2
B::B(int): 0
E::E(int, int):0, 4
B::~B(): 0
D::~D(): 1, 2
B::~B(): 1
E::~E():0, 4

d) B::B(int): 1
B::B(int): 0
D::D(int, int): 1, 2
E::E(int, int):0, 4
E::~E():0, 4
B::~B(): 0
D::~D(): 1, 2
B::~B(): 1

Answer: b)
Explanation: Always base class constructors are constructed followed by derived class constructors. Destruction will be in reverse order. Refer video lectures for more clarification.

Question 2

What will be the output of the following code ? *Mark 2*

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

class B { int id;
    public:
        static int count;
        B() { count++; id = count; cout << id << " ";}

};

class D : public B { int n;
    public:
        D(){count--; n = count; cout << n << " "; }
};

int B::count = 5;

int main() {

    B *basePtr = new D[2];
    delete [] basePtr;
    return 0;
}
```

Output of the Code is

a) 1 2 2 1

b) 1 2 3 4

c) 6 7 8 9

d) 6 5 6 5

Answer: d)
Explanation: Created array of objects of type `D`. Since, `B` is a base class, its constructor will be called first to increment the `count` value and later `D`'s constructor to decrement the value of `count`. Since, `count` is declared as `static`, it will remembers the value after every increment or decrement in both classes. Note: `count` is initialized with 6.

Question 3

Consider the following code segment. Assume that the `sizeof(int)= 4` and `sizeof(double)= 8`. What will be the size of the object `derived` ? *Mark 2*

```
class base {
    static int statInt;
    double arr[5];
    void display() { }
};

class base1: public base { };

class base2: public base { };

class derived: protected base1, protected base2 { };
```

a) 84

b) 120

c) 80

d) 44

Answer: c)
Explanation: Static members are not allocated with a memory in the class definition. Class `derived` ISA a base1 and base2 and hence inherits all its data and function members. But, memory will be allocated only for data members. Hence, the total memory will be 80 (5 (array elements) * 8 (double size) + 5 (array elements) * 8 (double size) = 80)

Question 4

Consider the following code segment. What will be the output for the following declarations in the main() function
Derived d(20, 30.5); d.print(); d.print(d); *Mark 2*

```
class Base {
   protected:
       int var;
   public:
       Base (int i) { var = i;}
       void print() { cout << var << " "; }
};

class Derived : protected Base { double dvar;
    public:
        Derived (int i, double d):Base(i), dvar(d) { }
        void print() {
            cout << var <<  " ";
        }

        void print(Derived& a) {
            cout << a.var << " " << a.dvar << " ";
        }
};
```

a) 20 20 30.5

b) 20 20

c) compilation error: print() cannot be overloaded in derived class

d) Compilation error: duplicate print() definitions

Answer: a)
Explanation: Print() function is both overrided and overloaded in derived class. Refer course video lectures for more detailed explanation.

Question 5

What will be the output of the following program ? *Mark 2*

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

class Department {
    public:
    string dept;
    Department(string d):dept(d) { }
    void getDeptName() { cout << dept;}
};

class Student : private Department {
  public:
    string name;
    Student(string n = "Mechanical", string d = "Electrical"):name(n),Department(d){}
    using Department::getDeptName;
};

int main() {

    Student s ("Civil");
    s.getDeptName();
    return 0;
}
```

a) Civil

b) Mechanical

c) Electrical

d) Compilation Error: getDeptName() cannot be accessed

Answer: c)
Explanation: Derived class contains a constructor with default parameters with initialization. The data member in the base class is initialized with the default argument. Since, all members of the base class are derived as private in derived class, the base class member function is accessed with `using`. Refer course video lectures for more details.

Programming Assignment

Question 1

Fill in the missing parts in the given program so that the given test cases will be satisfied. *Marks 3*

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

class Flower {
    public:
        string flwr;
        Flower(string f):flwr(f) { }
        void getFlowerName() { cout << flwr << " " << "is" << " " ; }
};

class Rose : ____________ { // Inherit Flower as private
    public:
        string name;

        /* Initialize name and flwr data members */
        Rose(string d = "Red", string n = "No flower") : _________, _________  { }
        void getFlowerName(Rose& r) { cout <<r.name << endl; }
        ___________________________ // Call base class getFlowerName() method
};


int main() {

    string flr, clor;
    cin >> flr >> clor;
    Rose s(flr);
    Rose r(flr, clor);
    s.getFlowerName();
    s.getFlowerName(r);
    return 0;
}
```

Answer: private Flower // name(n), Flower(d) // using Flower::getFlowerName;
Explanation: Base class needs to be inherited with the qualifier `private`. Since, base class is derived as private, initialize its data member with the name of base class. Call base member with `using` keyword.

a. Input:

```
rose red


Output:
rose is red
```

b. Input:

```
jasmine white
```

```
Output:
jasmine is white
```

c. Input:

```
apple green
```

```
Output:
apple is green
```

Question 2

Consider the following code. Fill up the code in editable section, with invocations of members of p, to match the outputs as given in the test cases. *Marks 3*

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

class M {
    protected:
        int m;
    public:
        void set_m(int x) { m = x; };
};

class N {
    protected:
        int n;
    public:
        void set_n(int y) { n = y; };
};

class P : public M, public N {
    public:
        void display() { cout << m * n << endl; };
};

int main() {
    int a, b;
    cin >> a >> b;
    P p;
//------------------------- Template Code (Editable) ---------------------
```

```
//------------------------- Suffix Fixed Code ---------------------------
return 0;
}
//-----------------------------------------------------------------------
```

Answer: p.set_m(a); // p.set_n(b); // p.display();
Explanation: The variables m and n are protected in base class. They will be available in the derive class. As per the test cases we want multiplication result of two inputs a and b. The multiplication happens in function display(). Hence it should be called. In turn it require m and n value which should be same as 'a' and 'b'. So we need to call p.set m(a) and p.set n(b)..

a. Input: 5 3

   `Output: 15`

b. Input: -20 10

   `Output: -200`

c. Input: -2 -4

   `Output: 8`

Question 3

Consider the following code. Fill up the code to match the outputs as given in the test cases. *Marks 2*

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

class Employee {
    public:
        string Name;
        double salary;
        Employee(string fName, double sal) : Name(fName), salary(sal) {}
        void show() {
            cout << Name << " " << salary;
        }

        void addBonus(double bonus) {
            salary += bonus;
        }
};

class Manager :public Employee {
```

```
    public:
        Manager(string fName, double sal) : Employee(fName, sal) {}
};

class Clerk :public Employee {
    public:
        Clerk(string fName, double sal) : Employee(fName, sal) {}
};

void congratulate(Employee* emp) {
    emp->addBonus(200);
    emp->show();
    cout << " ";
};

int main() {
    Employee* emp;
    int sal_m, sal_c;
    cin >> sal_c >> sal_m;
    Manager m1("Steve", sal_m);
    Clerk c1("Kevin", sal_c);
//------------------------- Template Code (Editable)-------------------------

// Write the code to congratulate the Manager and the Clerk

//------------------------- Suffix Fixed Code ---------------------------
return 0;
}
//-----------------------------------------------------------------------
```

Answer: congratulate(&c1); // congratulate(&m1);
Explanation: To congratulate the manager/clerk we have to call the function congratulate(). From the definition it is cleared that while calling the actual argument should be an address. Hence addresses of objects of manager and clerk are passed.

a. Input: 4000 2000

```
Output: Kevin 4200 Steve 2200
```

b. Input: 7200 2200

```
Output: Kevin 7400 Steve 2400
```

c. Input: 2100 1000

```
Output: Kevin 2300 Steve 1200
```