W3101 Programming Languages – C++ Midterm Feb 13, 2008

Name: Student Id:

---

1. Consider two classes, employee and executive such that executive class is derived from employee class,

```
class employee
{
  private:
    int salary;
  public:
    employee(int x) : salary(x)
      {
          cout << ‘‘In employee constructor’’ << endl;
      }
    ˜employee()
      {
          cout << ‘‘In employee destructor’’ << endl;
      }
  friend void setSalary (int amount, employee obj);
};

class executive : public employee
{
  public:
  // Write the constructor code here
  // This is the constructor
   executive(int amount) : employee (amount)
    {
        cout << ‘‘In executive constructor’’ << endl;
    }

    ˜executive()
      {
          cout << ‘‘In executive destructor’’ << endl;
      }
}
```

(a) Write the constructor code for the executive class. It should take an integer value called “amount” as input and set it as the executive’s salary. . . . (1 mark)

(b) What does the following code segment output? . . . (3 marks)

```
... // all the include files, etc.
void main()
{
    executive x(1);
    executive *y = new executive(2);
}
```

```
Answer:

 In employee constructor
 In executive constructor
 In employee constructor
 In executive constructor
 In executive destructor
 In employee destructor
```

(c) Define a function called “setSalary” that returns “void” (nothing) and takes an “int” and an object of employee as argument. Make it a friend function in the class employee. Write the body of the function, such that it sets the input parameter as the salary. . . . (2 marks)

```
void setSalary (int amount, employee obj)
{
  obj.salary = amount;
}
```

2. Consider the following code segment:

```
class baseClass
{
  public:
    baseClass() { };
    virtual void f1()
               { cout << ‘‘In base class f1’’ << endl; }
     void f2() { cout << ‘‘In base class f2’’ << endl; }
};

class derivedClass
{
  public:
    derivedClass(){ }
    void f1() { cout << ‘‘In derived class f1’’ << endl; }
    void f2() { cout << ‘‘In derived class f2’’ << endl; }
};
```

What is the ouput of the following program segment? Please write your answers next (or below) to the functions called in main. . . . (4 marks).

```
  main()
  {
    baseClass x;
    derivedClass y;
    baseClass *z = new derivedClass;

    x.f1();  // Answer: In base class f1

    x.f2();  // Answer: In base class f2

    y.f1();  // Answer: In derived class f1

    y.f2();  // Answer: In derived class f2

    z->f1(); // Answer: In derived class f1

    z->f2(); // Answer: In base class f2

  }
```

3.
```
    class employee
    {
      private:
       int salary;
      public:
       employee(..) { /* code */ }
     }
```

What are the different ways in which any external (not a member of this class) function or class access the private member of the class employee shown above? You are free to add member functions to the class, if you wish. Write one line code (to show how it is done) for each way you list in your answer. . . . (3 marks).

Answer: Any external class or function can access the private members using

(a) Public access methods — *e.g.*, <type> getMember() or
void setMember(<type> data)

(b) friend function — *e.g.*, friend void myFriendFunction (<type> args)

(c) friend class myFriendClass;

4. What is polymorphism? Is it really needed and if so, why? What are its advantages? How is it implemented in C++? If it makes your explanation easier, you can use an example. Be precise and brief, in your answers. . . . (2 marks).

   Answer: “Poly” means many, “morphism” means form. So, “polymorphism” is the property of an object to take many forms. That is, an object can behave as a derived class (*e.g.*, executive) or a derived class (*e.g.*, employee), depending on how it is used. It is an important property of object oriented programming. The advantage of polymorphism is that we can achieve the desired behaviour for many classes that are related to each other (by inheritance) which would otherwise be messy to implement.

   In C++, it is possible to implement the same function (with the same arguments) in two completely different ways in the base and derived classes. “Virtual functions” are an important means to achieve polymorphism. They can invoke the class specific functions, depending on how the class is created, than how it is declared, thus achieving polymorphism.