B.Sc. Engg. / HD CSE 3rd Semester

25 May 2018

ISLAMIC UNIVERSITY OF TECHNOLOGY (IUT)
ORGANISATION OF ISLAMIC COOPERATION (OIC)
Department of Computer Science and Engineering (CSE)

SEMESTER FINAL EXAMINATION — **WINTER SEMESTER, 2017-2018**

DURATION: 3 Hours — **FULL MARKS: 150**

CSE 4301: Object Oriented Programming

Programmable calculators are not allowed. Do not write anything on the question paper.

There are 8 (eight) questions. Answer any 6 (six) of them.

Figures in the right margin indicate marks.

---

1. a) Your development team at **Led-Labels** are trying to develop a multi-player first person shooting game. The multi-player feature of the game lets a number of players to connect over the network and to work as a team to kill zombies. Keeping up with the spirit of the team-work, the cumulative number of bullets that can be fired by the players in a game are fixed. There are a number of zombies who the players can shoot bullets at and kill. Moreover, a bullet can hit nothing but unimportant objects in the game. Finally, a bullet hitting a fellow player will 'kill' the player, forcing her to leave the game. However, a bullet hitting a zombie will get rid of it from the game. 25

Thus, the game consists of **GameObjects**. There are two specific types of such **GameObjects**, namely **Zombies** and **Players**. All **Zombies** and **Players** have their unique **IDs** represented as **integers** along with keeping track of the total number of **ammo**, **Zombies** and **Players** left in the game.

Now, your friend John Doe has already created a class called **Bullet** which has a member function called **whereDidItHit()**. The method returns **0** if a zombie is hit, **1** if a player is hit or **-1** if a generic game object is hit. Moreover, another method of **Bullet** class called **whoDidItHit()** returns the **ID** of the **GameObject** that was hit by the bullet.

Your job at **Led-Labels** is to at first complete the **GameObject**, **Zombie** and **Player** class. Later you need to device a class called **GameManager** which will include a method called **fire()**. The method will be called as long as there is ammo left in the game. All **Zombies** and **Players** in the game will initially subscribe themselves with the **GameManager** class by overloading **+=** operator. Each time a **fire()** event is published, all **Zombies** and **Players** will get notification of where the bullet had hit and if it had hit a specific **Zombie** or **Player**. The **Players** and **Zombies** will update the number of ammo, **Zombies** and **Players** left in the game according the result. If a specific **Zombie** or **Player** was hit by a bullet, they will unsubscribe from the **GameManager** by overloading **-=** operator. Finally, if all zombies become 'dead', the game will be won. On the other hand, if total ammo becomes depleted or only one player remains in the game, the game will be lost.

2. a) Consider the code in Figure 1. Write a code snippet to declare an array of 3 **Employees**. 5

```
class Employee {
 int salary, bonus;
public:
 Employee(int _salary) :salary(_salary), bonus(salary * 10 / 100) {}
 Employee(int _salary, int _bonus) :salary(_salary), bonus(salary*_bonus/10) {}
};
```

Figure 1: Code snippet for Question 2(a)

b) Write short notes on the following topics:

i. Function Overloading
ii. Virtual Base Class
iii. Virtual Function
iv. Abstract Class
v. Forward Declaration

c) Consider the code snippet in Question 2.(a). Now overload `<<` operator to set `salary` for an `Employee`. Also, overload `>>` operator to get the `salary` of an `Employee`. 10

3. a) Create a class called **StudentInfo** where the details of a student's academic records are stored. In the above-mentioned class, the private properties will include the name and ID of the student. Moreover, a student can take any number of courses and the results of the courses are to be kept in an integer array inside **StudentInfo** class. All these properties need to be private and can only be accessed outside of **StudentInfo** class by **CalculateAvgResult** method from **ResultService** class and through the parameterized constructor of the **StudentInfo** class. It should be noted that the properties of the **StudentInfo** class can only be set once through this its constructor as parameters and the values should be immutable. **CalculateAvgResult** method calculates the average result of a given student. Moreover, ResultService include another method called **SortStudents** which takes an array of students as parameter and sorts them according to their average results in ascending order. Finally, **SortStudents** prints the names of the students according to the sorted list. 20

Note: The **interface** and **implementation** for both the classes should be in separate files with each file marked elaborately along with their names. Moreover, you should include a main function in a separate file to demonstrate your implementation. Each file should refer to all required header and library files according to necessity.

b) What should be the output of the code snippet in Figure 2? 5

```
#include<iostream>
int x;
int& foo() {
    x = 30;
    return x;
}
```

```
int main() {
    int& (*fp) ();
    fp = foo;
    fp() = fp() + 43;
    std::cout << x << "\n";
    return 0;
}
```

Figure 2: Code Snippet for Question 3(b)

4. a) Simon and Garfunkel are software developers. Simon creates a class called `MyString` in `MyString.h` file. `MyString` class includes a character string as a private property. In the constructor of the `MyString` class, Simon takes input from the user for the string. Garfunkel wants to create a class called StringService inside a file called StringService.h. StringService class checks if a string is palindrome. Garfunkel wants to use the private variable in Simon's MyString class as test case for StringService class. However, Simon does not trust anyone in the world other than Garfunkel to give access to the private properties. 15

Your task is to recreate both these classes and provide Garfunkel access to Simon's private properties so that no one else can so the same.

b) What would be the output of the following code? 10

```
class Base1{
public:
    Base1(int x){cout << "In Base 1 Constructor. x = " << x << endl;}
    virtual void PrintLine(){cout << "Base1 function\n";}
    ~Base1(){cout << "Destroying Base1\n";}
};
```

```
class Base2{
public:
   Base2(int x){cout << "In Base 2 Constructor. x = " << x << endl;}
   virtual void PrintLine(){
      cout << "Base2 function\n";
      PrintAnotherLine();
   }
   virtual void PrintAnotherLine(){cout << "Another Line in Base\n";}
   ~Base2(){cout << "Destroying Base2\n";}
};

class Derived : public Base2,public Base1{
public:
    Derived() :Base1(10), Base2(20){}
    void PrintAnotherLine(){cout << "Another Line in Derived\n";}
};

int main(){
    Base1 *b1; Base2 *b2; Derived d;
    b1 = &d;
    b1->PrintLine();
    b2 = &d;
    b2->PrintLine();
    return 0;
}
```

Figure 3: Code snippet for Question 4(b)

5. a) **Cake** is a form of sweet dessert that is typically baked. Typical cake ingredients are flour, sugar, eggs, butter, oil, margarine etc. A cake requires specific amount of all these ingredients. It also contains a particular flavor. A **WeddingCake** is a specific type of cake that includes multiple tiers stacked one after another. On the other hand, a **BirthdayCake** contains a specific number of candles on top of it. A more recent type of cake is called **IUT_Cake** which is made every year on the eve of cake party at IUT. **IUT_Cake** contains all the properties of a **Cake** along with having multiple tiers like a **WeddingCake** and has candles on top like a **BirthdayCake**. Apart from these properties, **IUT_Cake** also has a special property – each cake is made for each year of fresher students of IUT. 15

Now, create four different classes to depict the object-oriented properties of the aforementioned types of cakes. What sort of problems are you going to face if you want to instantiate a new **IUT_Cake**? How can you solve that problem? Explain at least **two** different solutions.

b) Consider the following code segment: 10

```
class Base{
   int x;
public:
   Base():x(0){cout << "Default Constructor in Base. x = " << x << endl;}
   Base(int x):x(x){cout << "Parameterized Constructor in Base. x = "<<x<<endl;}
   virtual int get_base_X(){return x;}
};
class DerivedA : virtual public Base{
   int x;
public:
   DerivedA(int x,int y):x(x),Base(y){cout << "Constructor in DerivedA.x = " << x <<
endl;}
   virtual int get_base_X(){return x;}
};
class DerivedB : virtual public Base{
   int x;
public:
   DerivedB(int x, int y) :x(x),Base(y){
   cout << "Constructor in DerivedB. x = " << x << endl;}
   virtual int get_base_X(){return Base::get_base_X();}
```

```
};
class DerivedC:public DerivedB, public DerivedA{
   int x;
public:
   DerivedC(int x, int dax, int dbx, int bxa, int bxb) : x(x),DerivedA(dax, bxa),
DerivedB(dbx, bxb){
      cout << "Constructor in DerivedC. x = " << x <<endl;}
    int get_base_X(){return x;}
};
```

Figure 4: Code for Question 5(b)

Now write a main function such that it generates the following output:

```
Constructor in DerivedA. x = 2
Constructor in DerivedC. x = 1
Default Constructor in Base. x = 0
Constructor in DerivedB. x = 30
Constructor in DerivedA. x = 20
Constructor in DerivedC. x = 10
1
10

Parameterized Constructor in Base. x = -4
Constructor in DerivedA. x = -3
-3

Parameterized Constructor in Base. x = -40
Constructor in DerivedA. x = -30
-40
```

Figure 5: Output for Question 5(b)

Note: you can only create objects of the given classes, call different methods of those objects or print new lines in your implementation of main function. You can pass parameters to the objects.

6. a) Consider the following code snippet of a program. What are the reasons for the compilation errors in the program? 5

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

```
class derived_1 :private base {
public:    derived_1() { x = 1; y = 2; z = 3; }
};

class derived_2 :public derived_1 {
public:    derived_2() { x = 1; y = 2; z = 3; base b;  }
};
```

Figure 6: Output for Question 6(a)

b) What is run-time polymorphism and why is it necessary? 5

c) In question 2(a), **Employee** class is given. Now overload the **+** operator so that the chain addition of **Employee** objects and decimal values are possible. Let us assume, we have three **Employee** objects, **ob1**, **ob2** and **ob3**. Now, the overloaded operator should be able to execute the following expression: **10 + ob1 + ob2 + 12 + ob3**. 15

Note: When a decimal value is added to an **Employee** object, the value should be added to both **salary** and **bonus** of the object. On the other hand, when adding two **Employee** objects, their corresponding **salary** and **bonus** should be added respectively.

7. a) Consider the following code snippet of a program. The program is incomplete. You have to complete the program according to the instructions given in the code as comment. 20

```
class Course{
   int ID;
   string name;
   float GPA, credit_hour;
public:
   Course(){ /*Take input from the users to initialize the properties*/}
```

```
};
class CourseList{
   Course *cp;
   int course_list_size;
public:
   CourseList(){    /*Take the number of courses from user
             Dynamically allocate memory in 'cp'*/   }
   ~CourseList(){/*Release memory of 'cp'*/}
};
class Student{
   int ID;
   float CGPA;
   string name;
   CourseList course_list;
public:
   Student(){ /*Take input from the users to initialize the properties*/}
};
```

Figure 7: Code for Question 7(a)

Now implement the following operator overloading:

i. Overload += operator to add a course to a `CourseList`
ii. Overload = operator to set the value of `ID` for both `Student` and `Course` class
iii. Overload [] operator to get or set the GPA in a specific course from the `CourseList` based on the index passed as parameter.
iv. Calculate CGPA of a student by adding the GPAs of all courses and dividing the sum by the total number of courses.

b) Define the key characteristics of OOP. 5

8. a) What are the major bugs in the following code? How can you solve these bugs without changing any datatypes? 15

```
class Employee {
    int *salary;
public:
    Employee(int _salary) {
        this->salary = new int(_salary);
    }
    ~Employee() {
        delete this->salary;
    }
};
```

```
int main() {
    Employee ob(1);
    int inp;  cin >> inp;
    if (!inp % 2) {Employee ob1(0);
        ob1 = ob;
    }
    else {Employee ob2(0);
        ob2 = ob;
    }
    return 0;
}
```

Figure 8: Code for Question 8(a)

b) What is vector in C++. Give examples on how to use the following functions: `push_back()`, `begin()`, `end()`, `insert()`, `erase()`. 10