**CS-200 Fall 2014**

**LAB 10**

**Time Allowed: 2 Hours and 30 Minutes**

**Instructions:** You are allowed to discuss with your neighbors. But keep your voices down. Don’t standup and move around to discuss.

After lab you will have 72 hours to submit lab tasks on LMS with 40% deductions.

**INHERITANCE**

An important concept in object-oriented programming is inheritance. It provides a way for objects to define relationships with each other.

**What Is Inheritance?**

As the name inheritance suggests an object is able to inherit characteristics from another object. In more concrete terms, an object is able to pass on its state and behavior to its children. For inheritance to work, the objects need to have characteristics in common with each other. Classes can inherit attributes and behavior from pre-existing classes called base classes, super classes, parent classes or ancestor classes. The resulting classes are known as derived classes, subclasses or child classes. The relationships of classes through inheritance give rise to a hierarchy.

In our lab today, we will be using this concept of inheritance and establish a basic level hierarchy. Our hierarchy goes like this:



We have a HumanBeing class and clearly cricketers/phd-students are also human beings so they carry all the traits of human beings and **additionally** have some traits of their own. Hence we inherit the Cricketer and PhdStudent classes from HumanBeing class.

**How to show that a class is inherited?**

We show inheritance of a class in its declaration by referring to its parent class. Syntax for this is given as:

**Independent Class Declaration template:**

*class className{ }****;* <- (this semicolon :P)**

**Inherited Class Declaration template:**

*class className: public parentClassName {};*

**As a good programming practice we will do all our class declarations and class’s private/public methods/variables declarations in the header file, named ‘hierarchy.h’ today. Even the implementation of constructors must also be in ‘hierarchy.cpp’. Stick to these conventions, else you will be penalized.**

**Task 1:**

This introduction brings us to our very first task of the day. In this task you are required to do the basic declarations for the 3 given classes. Open ‘*hierarchy.h’* and have a look. You are given a basic structure of how this file should look like. From the declarations can you guess what the inheritance pattern is? Is the *HumanBeings* class inherited? Does Cricketer class inherit from *HumanBeings*? If yes, then how do you know? Does *PhdStudent* class inherit from *HumanBeings*? If you find any issues, then you must make appropriate changes to fix all this. If no, then continue.

Now you need to declare some private variables and public methods for all these classes. All human beings must have the following **abstracted** properties. They have a Name (string), a nationality (string) and age (int). All cricketers should have (in addition to all features of human beings) an average (float), maximum runs scored i.e maxScore (int) and total number of wickets taken i.e wicketsTaken (int). Similarly, all Phd students must have an advisorName (string), instituteName (string) and total number of research publications associated with him/her i.e numPublications (int) in addition to all a human being has.

**Note: In Cricketer and PhdStudent class, you only need to declare the ‘additional’ variables. As these classes are inherited, the ‘human being traits’ would automatically be imported.**

It’s now time to declare the constructors of Cricketer and PhdStudent class. The constructor should take in as argument all of their attributes, i.e from their attributes common with other human beings to their profession specific details. Constructors should look like:

*Cricketer(string name, string nation, int age, int maxScr, int wickets, float avg);*

*PhdStudent(string name, string nation, int age, string advisor, string univ ,int numPubls);*

Now that you have declared all the private variables, you now need to make some public set and get functions in the HumanBeing class, to access and modify their values. Required methods for other classes are already declared. You need to implement the following functions with exactly the same prototype.

*void setAge(int); //Set the age of human being*

*void setNation(string); //change nationality of a human being*

*String getName(); //get the name of human being*

*String getNation(); //get nationality of a human being*

Done with all the declarations, let’s move to the implementation part in hierarchy.cpp 

**Task 2 (Constructors and function Implementation)**

In this task you need to write all the constructor implementations first. Start off with the constructor of HumanBeing class. Write down its implementation. Now, for Cricketer and PhdStudent, we know they are inherited from HumanBeing class, we need to implement their constructors in a special way such that only the ‘additional’ features are assigned here and for the features inherited, we use the constructor of their parent (HumanBeing) class.

It works in the following way. Consider we have a parent class with the following constructor:

*ParentClass::ParentClass(int x , int y){*

*//Implementation*

*}*

then the child class would call its parent’s constructor in an appropriate way

*ChildClass::ChildClass(int x, int y , int z):ParentClass(x , y){*

*//Assigning the ‘additional’ feature z a value*

*}*

**Note: If even after reading the manual, going through the slides and reading the above template example, you are still confused, then call the TA and clarify your issue. Do not proceed without it.**

Now implement each and every function declared in ‘hierarchy.h’. You already know what all these functions are meant to do. After doing so, compile ‘main.cpp’. It should not produce any errors. Fix them, before you proceed.

**For the last two parts you have to create header (“.h”) and source (“.cpp”) files.**

**Task 3:**

Develop a program that contains two classes. The specification of both classes is as follow:

**Data** Class - Data class has following members protected String data - This field stores any type of user data. Data class has following member functions.

*One argument constructor which takes String object to initialize data.*

*public void setData(String d) - To set data.*

*public String getData() - To get data.*

*public String toString() - To override the toString() method of base*

*class Object. It simply returns data field.*

**Task 4:**

***EncryptedData*** Class - This class inherits from the Data class. This class has following data members

*protected String encryptedarray - Used to store encrypt data.*

*protected String secretKey – String that will be used to encrypt the data.*

*protected boolean encrypted - To show whether data is encrypted or not.*

*EncryptedData* class has following member functions

*One argument constructor which takes 2 String arguments. The first*

*initializes superclass Data, the second is the secretKey.*

*public void encrypt( ) - This function encrypts the data.*

*public void decrypt( ) - This function decrypts the already encrypted data.*

*public String printData() - This function should print out the encrypted Data if data is encrypted, otherwise should print the actual data.*

Encryption Technique:

To encrypt the data you will xor characters of the secretKey and data together. So the 1st character of the data will be xored with the 1st character of the key, 2nd character of the data with the 2nd character of the key and so on. Now the length of data might be greater than the length of the secretKey. In this case, when you get to the last character of the secretKey, the next character of the data will be *XORed* with the first character of the key. So for example If I have a scenario where the data is “**ABCDEFGH**” and the key is “**KEY**” you will *xor* the letters like so:

A with K, B with E, C with Y, D with K, F with E and so on...

To xor two characters togather use the ^ operator in C++. So if I want to XOR 'a' and 'b' I will do:

*char c = 'a' ^ 'b';*