Lab 4

Submit Assignment

Due Sunday by 11:59pm **Points** 100 **Submitting** a file upload **File Types** zip

OSU Information System

Goals

- Implement a program using inheritance and polymorphism
- Get more practice on classes and objects

In this lab, we will write an information system for Oregon State University. The purpose of this lab is to help you understand the basic concepts of inheritance and polymorphism.

Requirements

This program is a simple representation of an Oregon State information system that contains information about the university.

Below are the requirements for all the classes and the relationship between them.

Important: You may use vectors for this assignment if you choose so.

Note: For each class, you are required to create necessary functions such as **constructors, setters, getters, destructors** to make the program work.

University Class

The University class contains the following member variables:

- **name**: name of the university.

The name of the university MUST be “Oregon State University”

- **buildings**: contains n number of Building objects
- **people**: contains m number of Person objects

It contains the following member functions:

- A function that **prints the information on all the buildings** in its information system (name, address, and building’s size)

- A function that **prints the information of all the people** (name, age, GPA or Rating)

Note: The information on Building class and Person class are described below.

Note: When printing information of all people, you need to **distinguish** instructors and students by print either a “GPA” or “Rating”.

Building Class

The Building class contains the following member variables:

- **name**: name of the building
- **size**: the size of the building (in sqft)
- **address**: address of the building (stored as string)

You are encouraged to get the actual address and sizes of buildings that exist on the actual campus, here is the link to look this up, but this is not a hard requirement.

http://facilities.oregonstate.edu/buildings (http://facilities.oregonstate.edu/buildings)

Note: No additional member function required for this class.

Person Class

Person class is **polymorphic**, a Person pointer can hold an Instructor/Student object.

The Person class contains the following member variables:

- **name**: name of the person
- **age**: age of the person

The age of a person can be randomized or from input, but make it realistic

The Person class contains the following member function:

A function called “do_work()” that generates a random number that represents how many hours they will do work for, and then output message depending on if the Person is a Student or an Instructor.

Note: You can find more information on what message to output below.

Note: You can set a range for the random number to make the working hours look reasonable.

Student Class

The Student class contains the following member variable:

- **GPA**, the student’s GPA (double or float data type)

The range for GPA must be between 0.0 and 4.0.

For “do_work()” in Person class, if the Person is a Student, then the function should output the following message:

“PERSON_NAME did X hours of homework.” Where PERSON_NAME is the person’s name, and X is the randomly generated number.

Instructor Class

The Instructor class contains the following member variable:

- **rating**, the instructor’s rating (double or float data type)

The range for rating must be between 0.0 and 5.0.

For “do_work()” in Person class, if the Person is an Instructor, then the function should output the following message:

“PERSON_NAME graded papers for X hours.” Where PERSON_NAME is the person’s name, and X is the randomly generated number.

Menu

The menu must have **at least** the following options:

1. Prints information about all the buildings
2. Prints information of everybody at the university
3. Choose a person to do work
4. Exit the program

If option 3 is selected, the program should **print another menu** that prints all the people’s name and let the user input the choice of the person the user would like, to do work.

When the program starts, you need to **manually instantiate** at least 1 student, 1 instructor, and 2 buildings inside the University object. It can be done by either hard coding that information, or have the user input all the information at the start of the program.

You can add more options in the menu such as adding more Person/Buildings, or create other functions/variables to modularize code and complete the lab. It is not a hard requirement.

Style Guideline

Review the “Things not to do in the code” page and the “Things you need to do in your code” page as you will be held to those requirements. Make sure you indent the code correctly, and comment on code

appropriately.

Extra Credit 15%

Add options that have the following features:

- Save the information to a file
- Read a saved information from a file
- Add people to the program during runtime.

This will allow the user to add more people to the university, and close the program without losing all the information.

Note: This is an all or nothing extra credit. (There is no partial extra credit)

What you need to Submit

- All the program files including header and source files (.cpp/.hpp)
- Makefile

Important: Put all the files in a single .zip file and submit it on Canvas.

Grading

- Program Style/Comments -10%
- Correct classes have been created -30%
- Classes have correct member information -20%
- Class functions work correctly -20%
- Menu works correctly -20%

Some Rubric

Criteria	Ratings	Pts
Program Style Student has meaningful variable names, good commenting, organized program structure. No memory leaks.		10.0 pts
Classes Created Correctly University, Building, Person, Student and Instructor Classes created correctly. Person must be polymorphic and can point to either a Student or Instructor.		30.0 pts
Classes Have Correct Member Information		20.0 pts
Class Functions Work Correctly 1. Student and Instructor must override Person::do_work 2. University must have Person and Building print info functions.		20.0 pts
Menu Works Correctly Menus work correctly including input validation.		20.0 pts
Extra Credit (+ 15 pts) This is all or nothing: - Save info to file - Read info from saved file - Add people during runtime		0.0 pts
		Total Points: 100.0