

**FACULTY OF INFORMATICS**

|  |  |  |  |
| --- | --- | --- | --- |
| **SUBJECT’S INFORMATION:** | | | |
| Subject: | CSCI204 Object and Generic Programming | | |
| Session: | July 2014 | | |
| Programme / Section: | J766SENG (SE) / J766CS53 (MGD) / J766CS42 (DSS) | | |
| Lecturer: | Ms. Siti Hawa | | |
| Coursework Type  *(tick appropriate box)* | ❑ Individual Assignment ❑ Group Assignment ❑ Project  ✓Lab Task ❑ Seminar / Tutorial Paper ❑ Others | | |
| Coursework Title: | **Lab Task 4** | Coursework Percentage: | 1% |
| **ASSESSMENT CRITERIA:** | | | |
| Correctness | All programs should produce the correct result as stated in the specification. | | |
| Coding | Programs should use appropriate control structures and data structures correctly based on what have been covered in the class and stated in the specification. Necessary input validations should be done. | | |
| Readability | Appropriate comments are included. Meaningful identifiers used. Proper indentation and line spacing used. | | |
| Well formatted output | Output should be well formatted with appropriate messages displayed. Numbers are shown with appropriate precision. | | |
| **SUBMISSION:** | | | |
| All completed work should be submitted online through Moodle before or on the due date provided.  **SUBMIT AS EARLY AS POSSIBLE. YOU CAN RE-SUBMIT LATER IF NECESSARY. ONLY THE LATEST SUBMISSION WILL BE MARKED.**  **IF YOU SUBMIT YOUR ASSIGNMENT TWICE, ONE SUBMMISSION BEFORE THE DUE DATE AND ANOTHER AFTER THE DUE DATE, THEN YOU WILL BE PENALIZED FOR LATE SUBMISSON.** | | | |
| DUE DATE: | **WEEK 7** | | |
| **PENALTIES FOR LATE SUBMISSION:** | | | |
| Penalties apply to all late work, except if student academic consideration has been granted. Late submissions will attract a penalty of 25% of the assessment mark per day including the weekend. Work more than (3) days late will be awarded a mark of zero. | | | |
| **PLAGIARISM:** | | | |
| **When you submit an assessment task, you are declaring the following**   1. It is your own work and you did not collaborate with or copy from others. 2. You have read and understand your responsibilities under the University of Wollongong's policy on plagiarism. 3. You have not plagiarised from published work (including the internet). Where you have used the work from others, you have referenced it in the text and provided a reference list at the end ot the assignment.   Plagiarism will not be tolerated. Students are responsible for submitting original work for assessment, without plagiarising or cheating, abiding by the University’s policies on Plagiarism as set out in the University Handbook under University Policy Directory and in Faculty handbooks and subject guides. | | | |

**COURSEWORK SPECIFICATION**

**OBJECTIVES:**

In this lab task, you will experience how to write programs using friend functions and overloaded operators.

**TASK 1:**

Declare and implement two classes with the following definition:

1. The first class, Guest, holds a guest data, specifically a name and an identity card number.
2. The second, HotelRoom, holds hotel room information, specifically the hotel name, room number, and rate per day.
3. Each class should have a constructor that takes arguments to set the field values.
4. Create a friend function that displays the information of a guest staying in a hotel.

Write a short main() function to test the classes and the friend function.

**TASK 2:**

Write the implementation of a class called Length. The class should have:

1. Data fields associated with meter, centimeter, and millimeter.
2. Overload the extraction and insertion operators. Do not use a constructor to set field values.
3. Overload also the < () function to compare two Length objects and return true if the first object is less than the second object.
4. Define a member function that overloads the operator + () function used to add two Length object so that the following operation can be done:

Length length1, length2;

//some initialization takes place here

length1 = length1 + length2;

Write a main() function to test your class.

**TASK 3:**

Write code ScoreKeeper.cpp with class definition for ScoreKeeper. The class should store marks a student obtains in a subject. A different instance of the class is used for different subjects. The class should have:

1. Fields for the subject name, the number of scores per student and an array containing the list of the student’s scores on tests and assignments in class.
2. Write a constructor that accepts the subject name and number of assessment items. Then prompt the user for the marks based on the number of assessment given. Each mark should be between 0 and 100 inclusive, otherwise re-prompt.
3. Write an overloaded insertion operator for displaying the object data.
4. Write an overloaded = operator assigning one ScoreKeeper to another.

Write a main function to test your class.