

**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: | **Assignment 3** | Coursework Percentage: | 15% |
| **ASSESSMENT CRITERIA:** | | | |
| Correctness | All programs should produce the correct result as stated in the specification. | | |
| Coding | Proper data structure used. Good design of the program with complete functionalities included. Well-structured and used modular approached. Necessary data validation is implemented. | | |
| 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:** | | | |
| You are to submit through Moodle:   1. The softcopy of the project containing all files (.cpp and .h). 2. A word document containing the UML class diagram with explanation for each class and relationship.   **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: | **Friday, 21st November 2014, 5 pm** | | |
| **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:**

This assignment covers the following topics:

* Classes and objects
* Templates
* STL

**Remember that:**

**1. All programs should be able to run on the lab’s computers.**

**2. You must put the following information on the header of each text and source file you will be submitting in this assignment:**

**Student’s full name:**

**Student’s ID:**

**Modification Date:**

**Purpose of this file (or program):**

**3. Assignments that are not able to be compiled will result in zero mark given to the assignment.**

**4. You must only use the C++ features that have already been covered in the lectures**

**TASK 1:**

For this task, you are required to implement a container class called a Keyed\_Bag. A Keyed\_Bag is just like an actual bag such as a grocery bag where it keeps a collection of items in no particular order. However, each item added to the bag has a corresponding key.

The collection of the items in this container class should be kept as either an array or a linked list. Provide suitable data members for the class according to your choice. The class should have at least a default constructor to initialize an empty Keyed\_Bag.

Include a function to add an item to the container. The Keyed\_Bag container class should allow a data of an arbitrary type to be added by presenting a key (also of an arbitrary type). The type of the key and data may be different. Each item added must have a unique key but may have the same data. Include also a remove function to the container class. To remove a data from the Keyed\_Bag, the key of the data must be specified rather than the item itself. The function should return a Boolean value to indicate whether the removal is successful or not. Provide also the isEmpty() function and a destructor for your container class.

Implement a driver progam to demonstrate three different Keyed\_Bag objects. At least one of the Keyed\_Bag object should be used to store a user defined type for its data.

**TASK 2:**

For this task, you are required to write a program to simulate the process of assigning customers to an available table in a restaurant. Use the STL Deque to keep track of customers coming to the restaurant. Each customer will be assign to an available table on first come first serve basis. Each customer is given an id. On top of that a customer should also have the number of people coming with them to identify a suitable table to be assigned to them.

Use the STL map to keep a collection of tables in the restaurant. Each item in the map is identified by the table number. Each table keeps the information of the number of seats, customer (if not assigned, this value should be NULL), and a status to indicate if the table is occupied or not. There should not be any table with the same number.

When the program starts, populate the map with an initial number of tables. All tables should not be occupied upon initialization. You may use a text file with the following format for this initialization where each line contains the table number and number of seats available.

1 4

2 4

3 2

4 6

5 6

6 10

Declare and implement a simple Customer class with a customer id and total number of people as the data members. Include necessary constructor, accessor, and mutator functions for this class.

Use a menu driven program to simulate the arrival of customers, assignment of tables, adding new tables, and customer leaving the restaurant. When a customer arrives, dynamically create a customer object and push it to the deque containing the customers waiting to be assigned a table. A new customer should always be added to the end of the deque.

To assign a table, retrieve the first customer from the deque and search for an empty table in the map that has enough seats to assign the customer retrieved. If a table is available, remove the customer from the deque, assign the customer to the map item with the available table number, and set the status of the table to occupied. If no table is available, retrieve another customer record which might have a smaller number of people and search for an available table. Basically, customers are assigned with tables using the first come first serve basis. However, if there is no available table with enough seats, then the next customer with a smaller number can be assigned first. Use the iterator to traverse the deque for this purpose.

The user should allow for new tables to be added to the map. A new table added should always be initialized as an empty table with no customer assigned to it.

When a customer leaves the restaurant, search the map for the table number. If found, reset the customer to NULL and the status to unoccupied.

**MARKS ALLOCATION:**

|  |  |
| --- | --- |
| **Criteria (Task 1)** | **Marks Allocated** |
| Correctness | 1 |
| Collection class design and implementation | 3 |
| Main function | 1 |
| Readability and output formatting | 1 |
| **Total Marks for Task 1** | **6** |

|  |  |
| --- | --- |
| **Criteria (Task 2)** | **Marks Allocated** |
| Correctness | 1 |
| Use of STLs | 5 |
| Main function and other constructs | 2 |
| Readability and output formatting | 1 |
| **Total Marks for Task 2** | **9** |

**Total Marks for this Assignment is 15**