Week 2 Lab Assignment

Session: 2021 – 2025

Submitted by:

Ghulam Mustafa (2021-CS-39)

Supervised by:

Sir. Laeeq Khan Niazi

Department of Computer Science

University of Engineering and Technology Lahore Pakistan

OOP Task:

Implement one example of each concept

- Classes and Objects: Creating user-defined data types.
- Struct: Create user defined structure and store the data
- Inheritance: Reusing and extending functionality of base classes.
- Virtual Functions: For runtime polymorphism.

Screenshot:

```
E:\1 Study\7th Semester\CCL\Lab 2>OOP
Name: Mustafa Age: 20, Student ID: S123
Name: Saleem Age: 40 Subject: Math
```

Vectors Task:

You are tasked with developing a dynamic array-based solution to store an unknown number of student grades. Implement a program that allows insertion, deletion, and retrieval of elements in a vector. Demonstrate how the vector resizes dynamically when new elements are added beyond its current capacity.

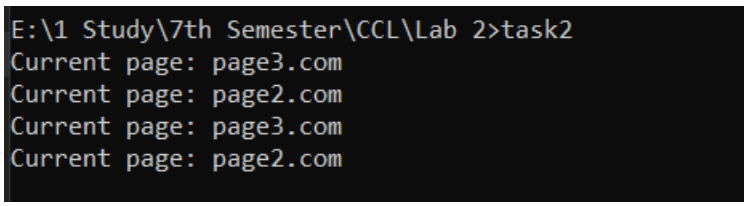
Screenshot:

```
E:\1 Study\7th Semester\CCL\Lab 2>g++ task1.cpp -o task1

E:\1 Study\7th Semester\CCL\Lab 2>task1
Grade is added to the vector array
Grade is added to the vector array
Grade is added to the vector array

The elements in array are: 8

The elements in array are: 6

The elements in array are: 7
The grade at the index: 2 will be removed which is: 7
The elements in array are: 8

The elements in array are: 6
The element found is: 6
E:\1 Study\7th Semester\CCL\Lab 2>
```

Lists: Doubly Task:

Create a doubly linked list to manage a sequence of webpages visited by a user in a browser. Implement functions for moving forward, backward, adding a new page, and deleting a page from the list. How would the design change if you were using a singly linked list instead?

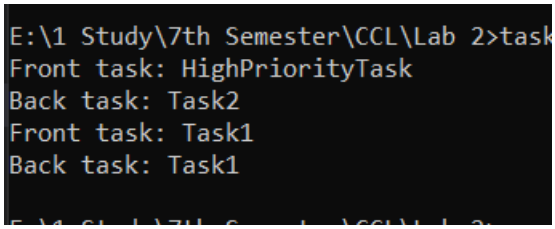
Screenshot:



Deque Task:

Implement a task scheduling system where tasks can be added from both the front and the back of a deque. Tasks with higher priority should be added at the front, while regular tasks should be added at the back. Demonstrate the operations of inserting, removing, and accessing elements from both ends.

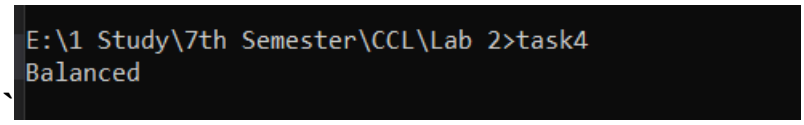
Screenshot:



Stack Task:

Design a program using a stack to check for balanced parentheses in a mathematical expression (e.g., {}, [], ()). Your solution should be able to handle expressions with nested parentheses and return true or false based on whether the expression is balanced.

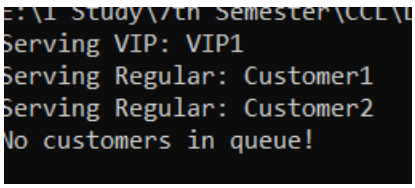
Screenshot:



Queue Task:

Implement a ticketing system for a cinema using a queue, where people are served in a first-come-first-served manner. Your program should allow customers to join the queue, process their tickets when they reach the front, and allow a VIP customer to be served at the next available opportunity.

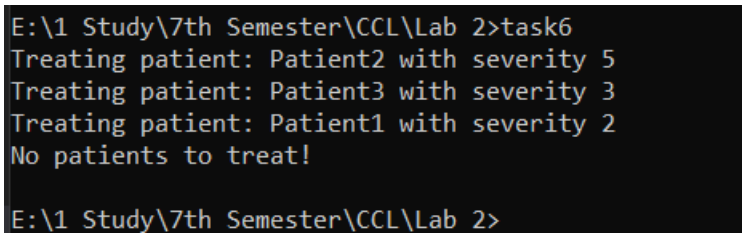
Screenshot:



Priority Queue Task:

You are building a hospital emergency room system where patients are attended based on the severity of their condition. Implement a priority queue where higher-severity patients are treated first, even if they arrive later than others with lower-severity conditions.

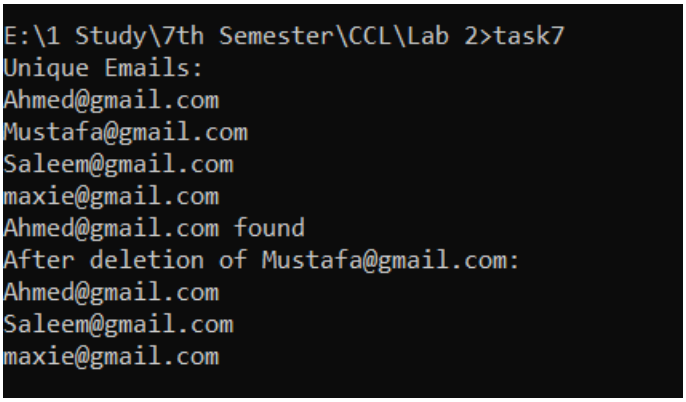
Screenshot:



Set Task:

Write a program to determine the unique elements in a list of customer email addresses. Use a set to eliminate duplicates and efficiently store the unique email addresses. Demonstrate set operations such as insertion, deletion, and searching.

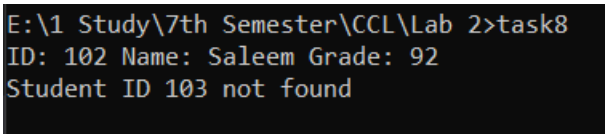
Screenshot:



Map Task:

Implement a student record management system using a map, where the student ID is the key and their details (name, grades, etc.) are stored as the value. The system should allow efficient retrieval, insertion, and deletion of student records by ID.

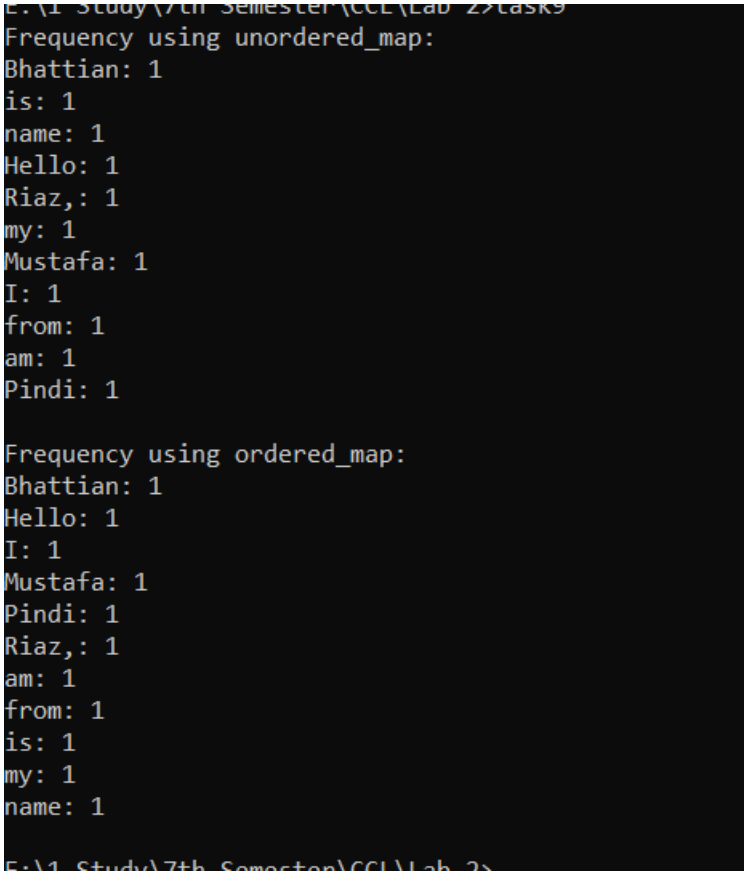
Screenshot:



Unordered Map Task:

Create a word frequency counter using an unordered map that counts how often each word appears in a given text. Compare the performance of this solution with an ordered map in terms of insertion and lookup time. When would you prefer to use an unordered map over a regular map?

Screenshot:



Multi-threading task:

Write a program that spawns two threads in C++: one thread prints numbers from 1 to 5, and the other thread prints numbers from 6 to 10. Use std::thread to run both threads concurrently.

Screenshot:

```
E:\1 Study\7th Semester\CCL\Lab 2>g++ Threading.cpp -o threading
Threading.cpp: In function 'void printNumbers1()':
Threading.cpp:10:14: error: 'std::this_thread' has not been declared
         std::this_thread::sleep_for(std::chrono::milliseconds(100));
              ^~~~~~~~~~~~
Threading.cpp: In function 'void printNumbers2()':
Threading.cpp:17:14: error: 'std::this_thread' has not been declared
         std::this_thread::sleep_for(std::chrono::milliseconds(100));
              ^~~~~~~~~~~~
Threading.cpp: In function 'int main()':
Threading.cpp:22:5: error: 'thread' is not a member of 'std'
     std::thread t1(printNumbers1);
     ^~~
Threading.cpp:23:5: error: 'thread' is not a member of 'std'
     std::thread t2(printNumbers2);
     ^~~
Threading.cpp:24:5: error: 't1' was not declared in this scope
     t1.join();
     ^~
Threading.cpp:25:5: error: 't2' was not declared in this scope
     t2.join();
     ^~
```