CC (lab2)

Session: 2021-2025

Submitted By:
Iqra Rafiq (2021-CS-134)

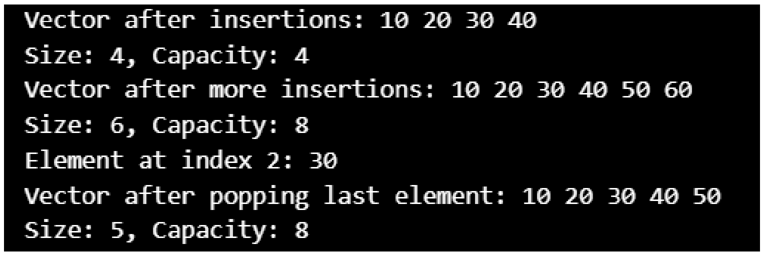
Supervised By:
Sir Laeeq Khan Niazi

Department of Computer Science
University of Engineering and Technology, Lahore
Pakistan

Problem 1

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

output:

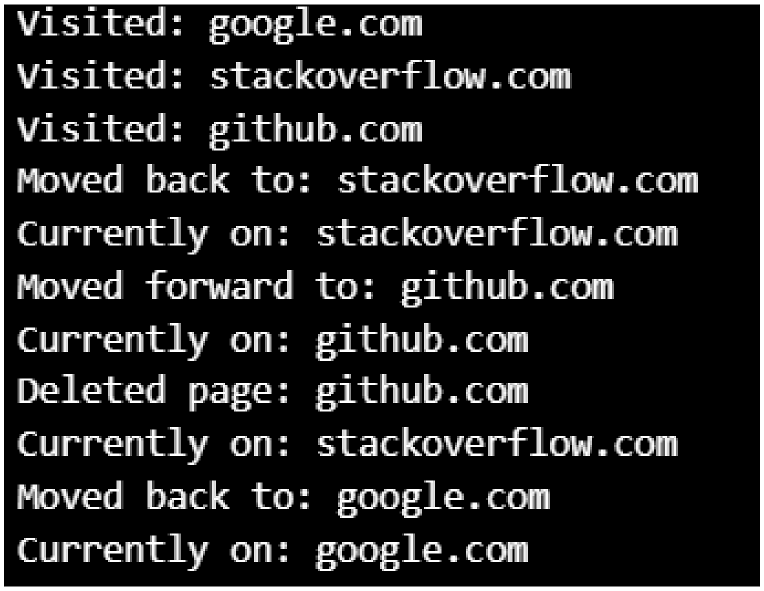


```
Vector after insertions: 10 20 30 40
Size: 4, Capacity: 4
Vector after more insertions: 10 20 30 40 50 60
Size: 6, Capacity: 8
Element at index 2: 30
Vector after popping last element: 10 20 30 40 50
Size: 5, Capacity: 8
```

Problem 2

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?

output:

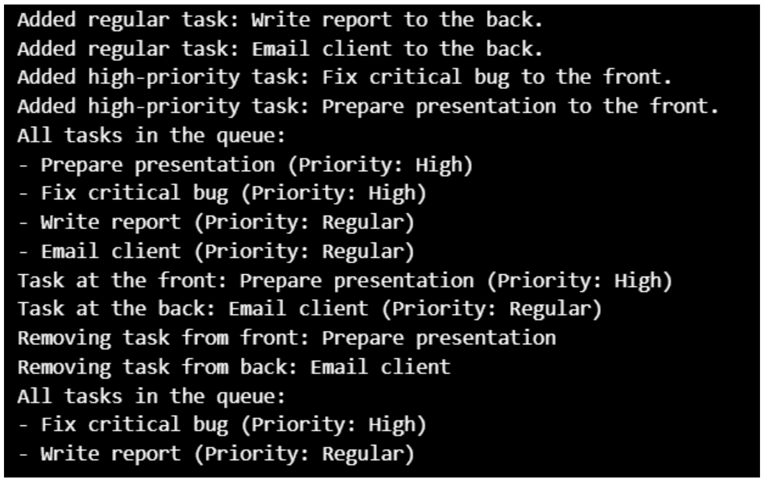


```
Visited: google.com
Visited: stackoverflow.com
Visited: github.com
Moved back to: stackoverflow.com
Currently on: stackoverflow.com
Moved forward to: github.com
Currently on: github.com
Deleted page: github.com
Currently on: stackoverflow.com
Moved back to: google.com
Currently on: google.com
```

Problem 3

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.

output:

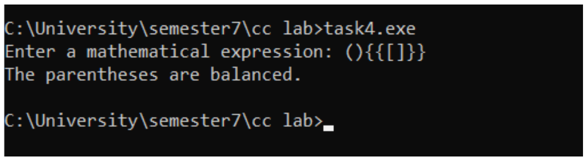


```
Added regular task: Write report to the back.
Added regular task: Email client to the back.
Added high-priority task: Fix critical bug to the front.
Added high-priority task: Prepare presentation to the front.
All tasks in the queue:
- Prepare presentation (Priority: High)
- Fix critical bug (Priority: High)
- Write report (Priority: Regular)
- Email client (Priority: Regular)
Task at the front: Prepare presentation (Priority: High)
Task at the back: Email client (Priority: Regular)
Removing task from front: Prepare presentation
Removing task from back: Email client
All tasks in the queue:
- Fix critical bug (Priority: High)
- Write report (Priority: Regular)
```

Problem 4

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

output:



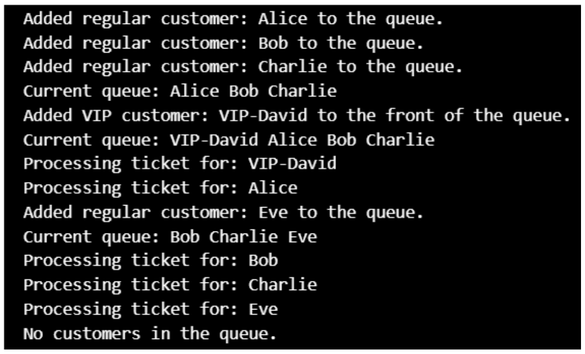
```
C:\University\semester7\cc lab>task4.exe
Enter a mathematical expression: (){{[]}}
The parentheses are balanced.

C:\University\semester7\cc lab>
```

Problem 5

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.

output:

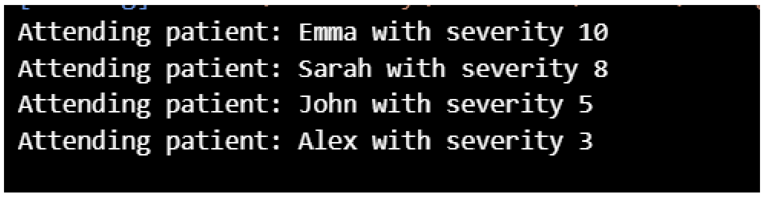


```
Added regular customer: Alice to the queue.
Added regular customer: Bob to the queue.
Added regular customer: Charlie to the queue.
Current queue: Alice Bob Charlie
Added VIP customer: VIP-David to the front of the queue.
Current queue: VIP-David Alice Bob Charlie
Processing ticket for: VIP-David
Processing ticket for: Alice
Added regular customer: Eve to the queue.
Current queue: Bob Charlie Eve
Processing ticket for: Bob
Processing ticket for: Charlie
Processing ticket for: Eve
No customers in the queue.
```

Problem 6

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.

output:

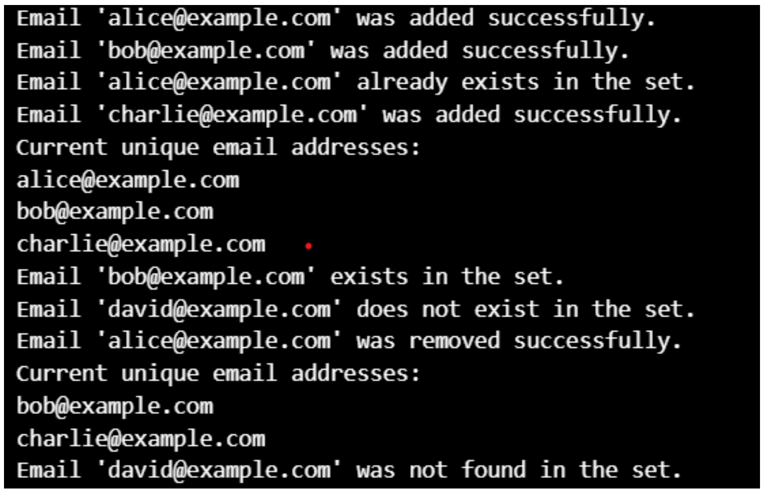


```
Attending patient: Emma with severity 10
Attending patient: Sarah with severity 8
Attending patient: John with severity 5
Attending patient: Alex with severity 3
```

Problem 7

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..

output:

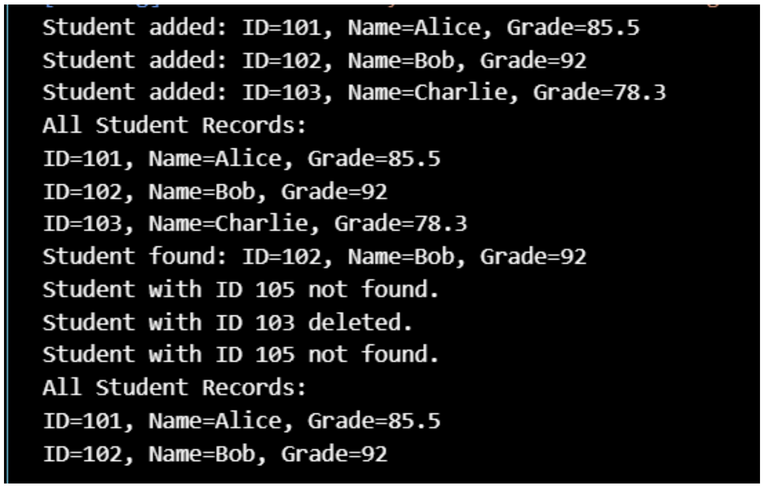


```
Email 'alice@example.com' was added successfully.
Email 'bob@example.com' was added successfully.
Email 'alice@example.com' already exists in the set.
Email 'charlie@example.com' was added successfully.
Current unique email addresses:
alice@example.com
bob@example.com
charlie@example.com
Email 'bob@example.com' exists in the set.
Email 'david@example.com' does not exist in the set.
Email 'alice@example.com' was removed successfully.
Current unique email addresses:
bob@example.com
charlie@example.com
Email 'david@example.com' was not found in the set.
```

Problem 8

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

output:

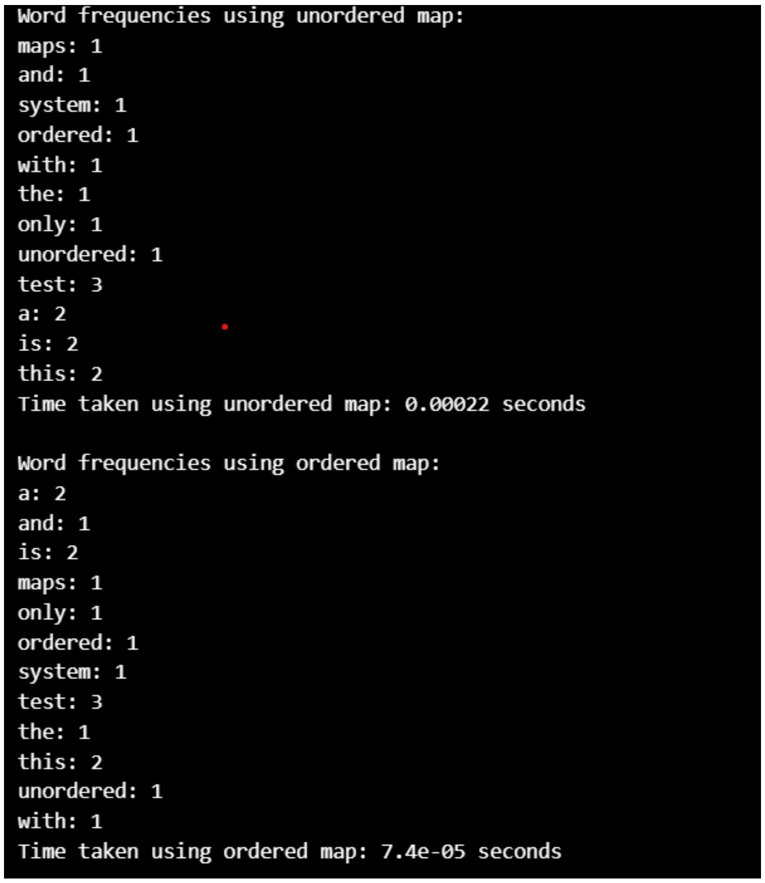


```
Student added: ID=101, Name=Alice, Grade=85.5
Student added: ID=102, Name=Bob, Grade=92
Student added: ID=103, Name=Charlie, Grade=78.3
All Student Records:
ID=101, Name=Alice, Grade=85.5
ID=102, Name=Bob, Grade=92
ID=103, Name=Charlie, Grade=78.3
Student found: ID=102, Name=Bob, Grade=92
Student with ID 105 not found.
Student with ID 103 deleted.
Student with ID 105 not found.
All Student Records:
ID=101, Name=Alice, Grade=85.5
ID=102, Name=Bob, Grade=92
```

Problem 9

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?

output:



```
Word frequencies using unordered map:
maps: 1
and: 1
system: 1
ordered: 1
with: 1
the: 1
only: 1
unordered: 1
test: 3
a: 2
is: 2
this: 2
Time taken using unordered map: 0.00022 seconds

Word frequencies using ordered map:
a: 2
and: 1
is: 2
maps: 1
only: 1
ordered: 1
system: 1
test: 3
the: 1
this: 2
unordered: 1
with: 1
Time taken using ordered map: 7.4e-05 seconds
```