UCS301 Data Structures

Lab Assignment 4 (Week 5)

Shefali Sharma

1024030284

Q1) Develop a menu driven program demonstrating the following operations on simple Queues: enqueue(), dequeue(), isEmpty(), isFull(), display(), and peek().

A1)

```
#include <iostream>
using namespace std;

#define MAX 5

class Queue {
  int arr[MAX];
  int front, rear;

public:
  Queue() {
    front = -1;
    rear = -1;
  }

  bool isFull() {
    return (rear == MAX - 1);
  }
```

```
bool isEmpty() {
    return (front == -1 || front > rear);
}

void enqueue(int x) {
    if (isFull()) {
        cout << "Queue Overflow!\n";
        return;
    }
    if (front == -1) front = 0;
    arr[++rear] = x;
    cout << x << " enqueued.\n";
}

void dequeue() {
    if (isEmpty()) {
        cout << "Queue Underflow!\n";
        return;
    }
    cout << arr[front++] << " dequeued.\n";
}

void peek() {
    if (isEmpty()) cout << "Queue is empty!\n";
    else cout << "Front element: " << arr[front] << endl;
}

void display() {
```

```
        if (isEmpty()) {
            cout << "Queue is empty!\n";
            return;
        }
        cout << "Queue elements: ";
        for (int i = front; i <= rear; i++)
            cout << arr[i] << " ";
        cout << endl;
    }
};

int main() {
    Queue q;
    int choice, val;

    do {
        cout << "\n--- Simple Queue Menu ---\n";
        cout << "1. Enqueue\n2. Dequeue\n3. Peek\n4. Display\n5. Exit\n";
        cout << "Enter choice: ";
        cin >> choice;

        switch (choice) {
            case 1: cout << "Enter value: "; cin >> val; q.enqueue(val); break;
            case 2: q.dequeue(); break;
            case 3: q.peek(); break;
            case 4: q.display(); break;
            case 5: cout << "Exiting...\n"; break;
            default: cout << "Invalid choice!\n";
```

```
    }
  } while (choice != 5);

  return 0;
}
```

```
--- Simple Queue Menu ---
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Exit
Enter choice: 4
Queue is empty!
```

2) Develop a menu driven program demonstrating the following operations on Circular Queues: enqueue(), dequeue(), isEmpty(), isFull(), display(), and peek().

A2)

```
#include <iostream>
using namespace std;

#define MAX 5

class CircularQueue {
  int arr[MAX];
  int front, rear;

public:
```

```
CircularQueue() {
   front = rear = -1;
}

bool isFull() {
   return (front == 0 && rear == MAX - 1) || (front == rear + 1);
}

bool isEmpty() {
   return (front == -1);
}

void enqueue(int x) {
   if (isFull()) {
      cout << "Circular Queue Overflow!\n";
      return;
   }
   if (front == -1) front = 0;
   rear = (rear + 1) % MAX;
   arr[rear] = x;
   cout << x << " enqueued.\n";
}

void dequeue() {
   if (isEmpty()) {
      cout << "Circular Queue Underflow!\n";
      return;
   }
```

```
        cout << arr[front] << " dequeued.\n";
        if (front == rear) front = rear = -1;
        else front = (front + 1) % MAX;
    }

    void peek() {
        if (isEmpty()) cout << "Circular Queue is empty!\n";
        else cout << "Front element: " << arr[front] << endl;
    }

    void display() {
        if (isEmpty()) {
            cout << "Circular Queue is empty!\n";
            return;
        }
        cout << "Circular Queue elements: ";
        int i = front;
        while (true) {
            cout << arr[i] << " ";
            if (i == rear) break;
            i = (i + 1) % MAX;
        }
        cout << endl;
    }
};

int main() {
    CircularQueue cq;
```

```
    int choice, val;

    do {
        cout << "\n--- Circular Queue Menu ---\n";
        cout << "1. Enqueue\n2. Dequeue\n3. Peek\n4. Display\n5. Exit\n";
        cout << "Enter choice: ";
        cin >> choice;

        switch (choice) {
            case 1: cout << "Enter value: "; cin >> val; cq.enqueue(val); break;
            case 2: cq.dequeue(); break;
            case 3: cq.peek(); break;
            case 4: cq.display(); break;
            case 5: cout << "Exiting...\n"; break;
            default: cout << "Invalid choice!\n";
        }
    } while (choice != 5);

    return 0;
}
```

```
--- Circular Queue Menu ---
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Exit
Enter choice: 1
Enter value:
10
10 enqueued.

--- Circular Queue Menu ---
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Exit
Enter choice: 3
Front element: 10
```

3) Write a program interleave the first half of the queue with second half. Sample I/P: 4 7 11 20 5 9 Sample O/P: 4 20 7 5 11 9

A3)

```
#include <iostream>

#include <queue>

using namespace std;


void interleave(queue<int>& q) {

   int n = q.size();

   int half = n / 2;
```

```
    queue<int> firstHalf;

    for (int i = 0; i < half; i++) {
        firstHalf.push(q.front());
        q.pop();
    }

    while (!firstHalf.empty()) {
        q.push(firstHalf.front());
        firstHalf.pop();
        q.push(q.front());
        q.pop();
    }
}

int main() {
    queue<int> q;
    int n, val;
    cout << "Enter number of elements: ";
    cin >> n;
    cout << "Enter elements: ";
    for (int i = 0; i < n; i++) {
        cin >> val;
        q.push(val);
    }

    interleave(q);
```

```
    cout << "Interleaved queue: ";
    while (!q.empty()) {
        cout << q.front() << " ";
        q.pop();
    }
    cout << endl;

    return 0;
}
```

Output

```
Enter number of elements: 3
Enter elements: 10 20 30
Interleaved queue: 30 10 20


=== Code Execution Successful ===
```

4) Write a program to find first non-repeating character in a string using Queue. Sample I/P: a a b c Sample O/P: a -1 b b

A4)

```
#include <iostream>
#include <queue>
#include <unordered_map>
using namespace std;
```

```
void firstNonRepeating(string s) {
    unordered_map<char, int> freq;
    queue<char> q;

    for (char ch : s) {
        freq[ch]++;
        q.push(ch);

        while (!q.empty() && freq[q.front()] > 1)
            q.pop();

        if (q.empty()) cout << -1 << " ";
        else cout << q.front() << " ";
    }
    cout << endl;
}

int main() {
    string s;
    cout << "Enter string: ";
    cin >> s;

    firstNonRepeating(s);

    return 0;
}
```

Output

```
Enter string: SHEFALI
S S S S S S S


=== Code Execution Successful ===
```

5) Write a program to implement a stack using (a) Two queues and (b) One Queue.

A5)

```
#include <iostream>
#include <queue>
using namespace std;

class Stack {
  queue<int> q1, q2;

public:
  void push(int x) {
    q2.push(x);
    while (!q1.empty()) {
      q2.push(q1.front());
      q1.pop();
    }
```

```
    swap(q1, q2);
}

void pop() {
    if (q1.empty()) {
        cout << "Stack Underflow!\n";
        return;
    }
    cout << q1.front() << " popped.\n";
    q1.pop();
}

void top() {
    if (q1.empty()) cout << "Stack is empty!\n";
    else cout << "Top element: " << q1.front() << endl;
}

void display() {
    if (q1.empty()) {
        cout << "Stack is empty!\n";
        return;
    }
    cout << "Stack elements: ";
    queue<int> temp = q1;
    while (!temp.empty()) {
        cout << temp.front() << " ";
        temp.pop();
    }
```

```
        cout << endl;
    }
};

int main() {
    Stack st;
    st.push(10);
    st.push(20);
    st.push(30);
    st.display();
    st.pop();
    st.top();
    st.display();
    return 0;
}
```

Output

```
Stack elements: 30 20 10
30 popped.
Top element: 20
Stack elements: 20 10


=== Code Execution Successful ===
```