Riphah International University

Data Structure (Lab)

Lab Task 5th

Name: Mir Ahmad Shah

Sap id: 54906

Instructor: Mr. Zeeshan Ali

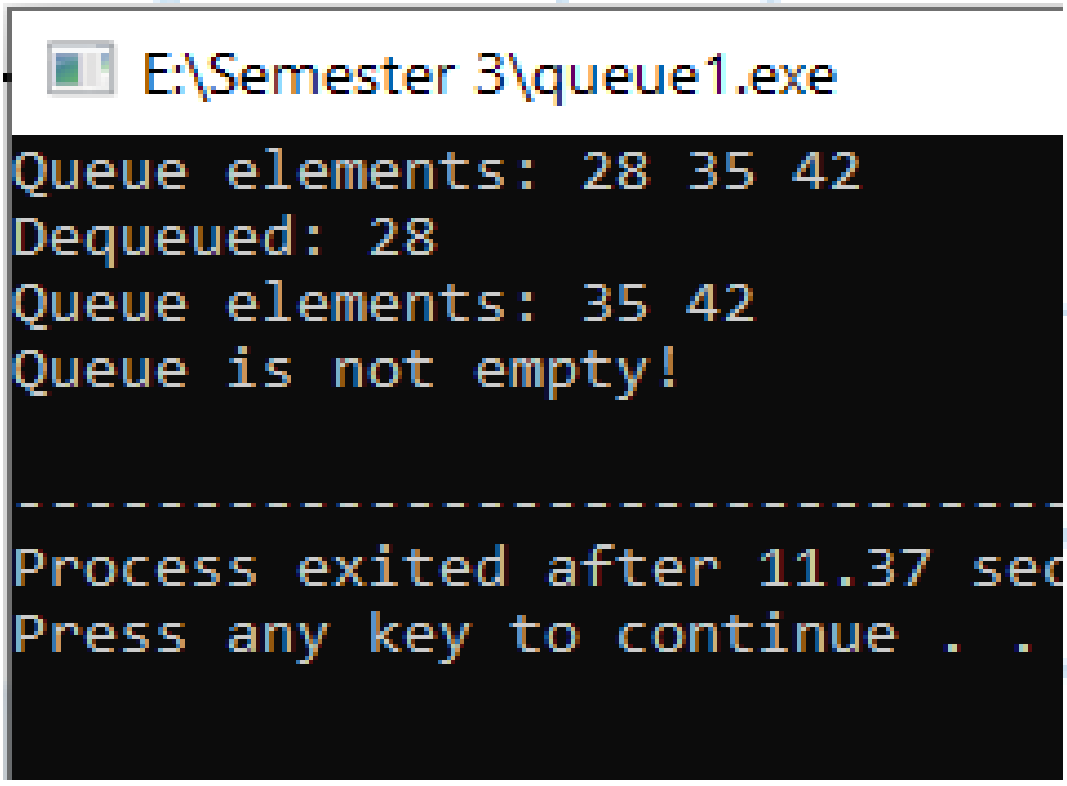
Section: BSCS (3-1)

Submission Date: September 20, 2024

1:

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

class Queue {
private:
    int* queueArray;
    int capacity;
    int front;
    int rear;
    int size;

public:
    Queue(int capacity = 100) {
        this->capacity = capacity;
        queueArray = new int[capacity];
        front = 0;
        rear = -1;
        size = 0;
    }

    void enqueue(int dataItem) {
        if (size >= capacity) {
            cout << "Queue overflow, cannot add more elements!" << endl;
            return;
        }
        rear = (rear + 1) % capacity;
        queueArray[rear] = dataItem;
        size++;
    }

    int dequeue() {
        if (isEmpty()) {
            cout << "Queue underflow, no elements to dequeue!" << endl;
            return -1;
        }
        int item = queueArray[front];
        front = (front + 1) % capacity;
        size--;
        return item;
    }

    bool isEmpty() {
        return (size == 0);
    }

    void display() {
        if (isEmpty()) {
            cout << "Queue is empty!" << endl;
            return;
        }
```

```
        cout << "Queue elements: ";
        for (int i = 0; i < size; i++) {
            cout << queueArray[(front + i) % capacity] << " ";
        }
        cout << endl;
    }
};

int main() {
    Queue myQueue;

    myQueue.enqueue(28);
    myQueue.enqueue(35);
    myQueue.enqueue(42);

    myQueue.display();
    cout << "Dequeued: " << myQueue.dequeue() << endl;

    myQueue.display();

    if (myQueue.isEmpty()) {
        cout << "Queue is empty!" << endl;
    } else {
        cout << "Queue is not empty!" << endl;
    }

    return 0;
}
```



2:

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

class Queue {
private:
    char* queueArray;
    int capacity;
    int front;
    int rear;
    int size;

public:
    Queue(int capacity = 100) {
        this->capacity = capacity;
        queueArray = new char[capacity];
        front = 0;
        rear = -1;
        size = 0;
    }
    void enqueue(char dataItem) {
        if (size >= capacity) {
            cout << "Queue overflow, cannot add more elements!" << endl;
            return;
        }
        rear = (rear + 1) % capacity;
        queueArray[rear] = dataItem;
        size++;
    }

    char dequeue() {
        if (isEmpty()) {
            cout << "Queue underflow, no elements to dequeue!" << endl;
            return '\0';
        }
        char item = queueArray[front];
        front = (front + 1) % capacity;
        size--;
        return item;
    }

    bool isEmpty() {
        return (size == 0);
    }
```

```
    void display() {
        if (isEmpty()) {
            cout << "Queue is empty!" << endl;
            return;
        }
        cout << "Queue elements: ";
        for (int i = 0; i < size; i++) {
            cout << queueArray[(front + i) % capacity];
            if (i < size - 1)
                                cout << " > ";
        }
        cout << endl;
    }

    void concatenate(Queue& other) {
        for (int i = 0; i < other.size; i++) {
            enqueue(other.queueArray[(other.front + i) % other.capacity]);
        }
    }
};

int main() {
    const int maxLength = 100;
    char input[maxLength];
    cout << "Enter a string: ";
    cin.getline(input, maxLength);

    const int maxWords = 100;
    Queue queues[maxWords];
    int wordCount = 0;

    int i = 0;
    while (input[i] != '\0') {
        if (input[i] != ' ' && (i == 0 || input[i - 1] == ' ')) {
            while (input[i] != ' ' && input[i] != '\0') {
                queues[wordCount].enqueue(input[i]);
                i++;
            }
            wordCount++;
        } else {
            i++;
        }
    }

    Queue finalQueue;
```

```
    for (int j = 0; j < wordCount; j++) {
        finalQueue.concatenate(queues[j]);
    }

    cout << "Concatenated Queue: ";
    finalQueue.display();

    return 0;
}
```