MUHAMMAD SARMAD CHUGHTAI
54915 CS3-1
DATA STRUCTURE

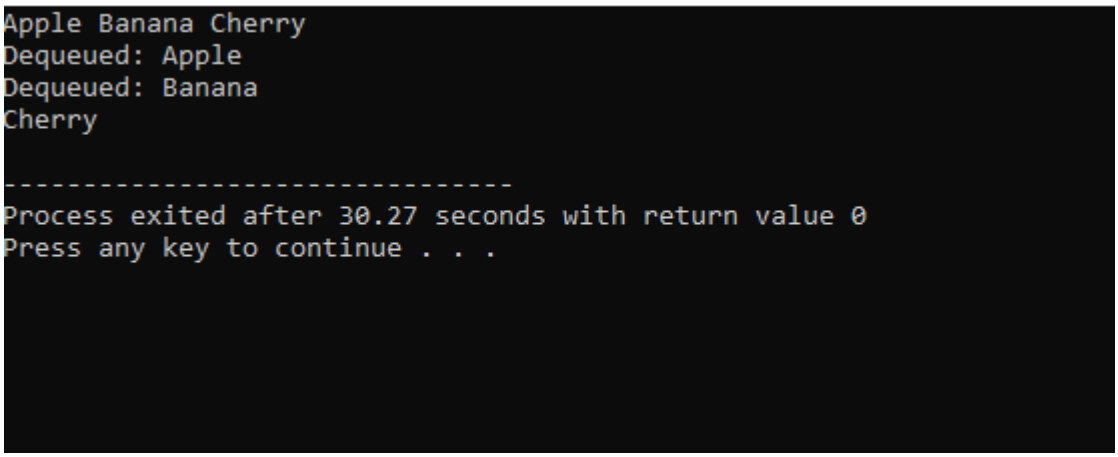
Lab Task 05

Question no 1:

```
#include <iostream>

using namespace std;

class Queue {
private:
    int front;
    int rear;
    int capacity;
    string* data;

public:
    Queue(int capacity = 100) : capacity(capacity), front(0), rear(0) {
        data = new string[capacity];
    }

    ~Queue() {
        delete[] data;
    }

    bool enqueue(const string& element) {
        if ((rear + 1) % capacity == front) {
            cout << "Error: Queue overflow!" << endl;
            return false;
        }
        data[rear] = element;
        rear = (rear + 1) % capacity;
        return true;
    }

    string dequeue() {
        if (front == rear) {
            cout << "Error: Queue underflow!" << endl;
            return "";
        }
        string element = data[front];
        front = (front + 1) % capacity;
        return element;
    }

    bool isEmpty() {
        return front == rear;
    }

    void display() {
        for (int i = front; i != rear; i = (i + 1) % capacity) {
            cout << data[i] << " ";
        }
```

```
        cout << endl;
        return;
    }
};

int main() {
    Queue q;

    q.enqueue("Apple");
    q.enqueue("Banana");
    q.enqueue("Cherry");

    q.display(); // Output: Apple Banana Cherry

    cout << "Dequeued: " << q.dequeue() << endl; // Output: Apple
    cout << "Dequeued: " << q.dequeue() << endl; // Output: Banana

    q.display(); // Output: Cherry

    return 0;
}
```

Output :



Question no 2:

```
#include <iostream>

using namespace std;

class Queue {
private:
    int front;
    int rear;
    int capacity;
    string* data;

public:
    Queue(int capacity = 100) : capacity(capacity), front(0), rear(0) {
        data = new string[capacity];
    }

    ~Queue() {
        delete[] data;
    }

    bool enqueue(const string& element) {
        if ((rear + 1) % capacity == front) {
```

```
            cout << "Error: Queue overflow!" << endl;
            return false;
        }
        data[rear] = element;
        rear = (rear + 1) % capacity;
        return true;
    }

    string dequeue() {
        if (front == rear) {
            cout << "Error: Queue underflow!" << endl;
            return "";
        }
        string element = data[front];
        front = (front + 1) % capacity;
        return element;
    }

    bool isEmpty() {
        return front == rear;
    }

    void display() {
        for (int i = front; i != rear; i = (i + 1) % capacity) {
            cout << data[i] << " ";
        }
        cout << endl;
        return;
    }
};

int main() {
    string input;
    cout << "Enter a string: ";
    getline(cin, input);

    Queue queues[100];
    int queueIndex = 0;
    string word;
    int wordIndex = 0;

    for (int i = 0; i < input.length(); i++) {
        if (input[i] == ' ') {
            word = input.substr(wordIndex, i - wordIndex);
            queues[queueIndex].enqueue("Q" + to_string(queueIndex + 1) + "-" + word);
            queueIndex++;
            wordIndex = i + 1;
        }
    }

    word = input.substr(wordIndex, input.length() - wordIndex);
    queues[queueIndex].enqueue("Q" + to_string(queueIndex + 1) + "-" + word);

    Queue result;
    for (int i = 0; i <= queueIndex; i++) {
        while (!queues[i].isEmpty()) {
            result.enqueue(queues[i].dequeue());
        }
    }

    cout << "Concatenated queue: ";
    result.display();
```

```
    return 0;
}
```

Output :