

**Data Structures**

**(Lab)**

**Task 5**

**Name:** Muqeet Ahmed

**SAP ID:** 53102

**Semester:** 3rd

**Q1:**

#include <iostream>

using namespace std;

class Queue {

private:

static const int SIZE = 100;

int front, rear;

int arr[SIZE];

public:

Queue() {

front = -1;

rear = -1;

}

bool isEmpty() {

return (front == -1 && rear == -1);

}

bool isFull() {

return (rear == SIZE - 1);

}

void enqueue(int value) {

if (isFull()) {

cout << "Queue is full. Cannot enqueue." << endl;

return;

} else if (isEmpty()) {

front = rear = 0;

} else {

rear++;

}

arr[rear] = value;

cout << "Enqueued: " << value << endl;

}

void dequeue() {

if (isEmpty()) {

cout << "Queue is empty. Cannot dequeue." << endl;

return;

} else if (front == rear) {

front = rear = -1;

} else {

cout << "Dequeued: " << arr[front] << endl;

front++;

}

}

void display() {

if (isEmpty()) {

cout << "Queue is empty." << endl;

return;

}

cout << "Queue elements: ";

for (int i = front; i <= rear; i++) {

cout << arr[i] << " ";

}

cout << endl;

}

};

int main() {

Queue q;

q.enqueue(10);

q.enqueue(20);

q.enqueue(30);

q.display();

q.dequeue();

q.display();

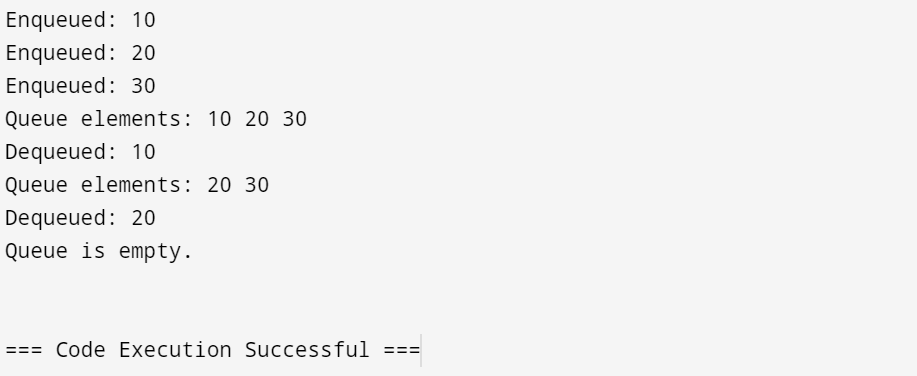
q.dequeue();

q.dequeue();

q.display();

return 0;

}



**Q2:**

#include <iostream>

#include <queue>

#include <sstream>

#include <vector>

using namespace std;

void printQueue(queue<char> q) {

while (!q.empty()) {

cout << q.front() << " ";

q.pop();

}

cout << endl;

}

queue<char> concatenateQueues(const vector<queue<char>>& queues) {

queue<char> finalQueue;

for (auto q : queues) {

while (!q.empty()) {

finalQueue.push(q.front());

q.pop();

}

}

return finalQueue;

}

int main() {

string input;

cout << "Enter a string: ";

getline(cin, input);

vector<queue<char>> queues;

stringstream ss(input);

string word;

while (ss >> word) {

queue<char> q;

for (char c : word) {

q.push(c);

}

queues.push\_back(q);

}

for (int i = 0; i < queues.size(); i++) {

cout << "Queue " << i + 1 << ": ";

printQueue(queues[i]);

}

queue<char> finalQueue = concatenateQueues(queues);

cout << "Final concatenated queue: ";

printQueue(finalQueue);

return 0;

}

