**Foundation University**

**School of Science and Technology**



**Department of Engineering Technology**

**Data Structure Lab Report : 5**

**Name : Wazzaif Ali**

**Roll no : 095**

## Exercises:

After every dequeue operation, check if the queue becomes empty, and reset both front Implement a Circular Queue in C++ using an array. The queue should be initialized with a size between 3 and 7. Ensure the following:

1. and rear pointers.
2. The queue should automatically resize to double its capacity when it becomes full and print a message indicating the resize.
3. After resizing, demonstrate that the queue continues to work as expected.

## Solution:

#include <iostream>

using namespace std;

class CircularQueue { private: int \*arr; int front; int rear; int size;

int count;

public:

CircularQueue(int s) { if (s < 3) s = 3; else if (s > 7) s = 7; size = s; arr = new int[size]; front = -1; rear = -1; count = 0;

cout << "Queue initialized with size: " << size << endl;

}

bool isFull() {

return count == size;

}

bool isEmpty() {

return count == 0;

}

void enqueue(int value) {

if (isFull()) { resize();

}

if (front == -1) front = 0; rear = (rear + 1) % size; arr[rear] = value;

count++;

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

}

void dequeue() {

if (isEmpty()) {

cout << "Queue is empty! Nothing to dequeue.\n"; return;

}

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

front = (front + 1) % size;

count--; if (count == 0) { front = rear = -1;

cout << "Queue is now empty. Resetting front and rear.\n";

}

}

void resize() { int newSize = size \* 2; int \*newArr = new int[newSize];

cout << "Queue is full. Resizing to " << newSize << "...\n"; for (int i = 0; i < count; i++) {

newArr[i] = arr[(front + i) % size];

}

delete[] arr; arr = newArr; size = newSize; front = 0;

rear = count - 1;

}

void display() { if (isEmpty()) { cout << "Queue is empty.\n";

return;

}

cout << "Queue elements: "; for (int i = 0; i < count; i++) {

cout << arr[(front + i) % size] << " ";

}

cout << endl;

}

~CircularQueue() {

delete[] arr;

}

};

int main() { CircularQueue q(4);

q.enqueue(25);

q.enqueue(35);

q.enqueue(20);

q.enqueue(50);

q.display();

q.enqueue(60);

q.display();

q.dequeue();

q.dequeue();

q.display();

q.enqueue(80);

q.enqueue(90);

q.display();

q.dequeue();

q.dequeue();

q.dequeue();

q.dequeue();

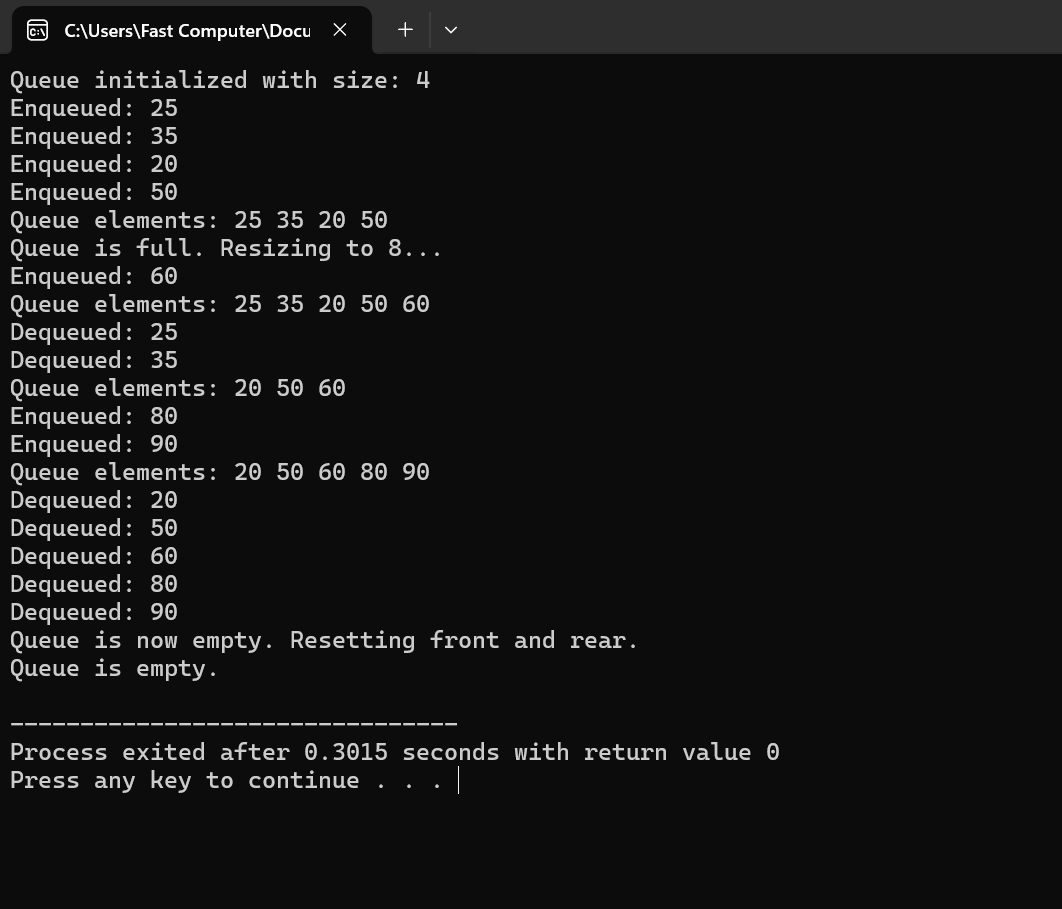
q.dequeue();

q.display();

return 0;

}

**Output:**

****