|  |  |
| --- | --- |
| ASSIGNMENT 4 SOLUTIONS | Ishaan Mishra  1024030346 |

# Lab Assignment 4: Queues (C++ Solutions)

## Q1, Q2. Simple & Circular Queue (menu-driven)

#include <iostream>  
using namespace std;  
#define MAX 5  
  
class Queue {  
 int arr[MAX], front, rear;  
public:  
 Queue() { front = rear = -1; }  
  
 bool isEmpty() { return front == -1; }  
 bool isFull() { return (rear + 1) % MAX == front; }  
  
 void enqueue(int x) {  
 if (isFull()) { cout << "Queue is Full\n"; return; }  
 if (front == -1) front = 0;  
 rear = (rear + 1) % MAX;  
 arr[rear] = x;  
 }  
  
 void dequeue() {  
 if (isEmpty()) { cout << "Queue is Empty\n"; return; }  
 cout << "Dequeued: " << arr[front] << endl;  
 if (front == rear) front = rear = -1;  
 else front = (front + 1) % MAX;  
 }  
  
 void peek() {  
 if (!isEmpty()) cout << "Front: " << arr[front] << endl;  
 else cout << "Queue Empty\n";  
 }  
  
 void display() {  
 if (isEmpty()) { cout << "Queue Empty\n"; return; }  
 int i = front;  
 cout << "Queue: ";  
 while (true) {  
 cout << arr[i] << " ";  
 if (i == rear) break;  
 i = (i + 1) % MAX;  
 }  
 cout << endl;  
 }  
};  
  
int main() {  
 Queue q;  
 int choice, val;  
 while (1) {  
 cout << "1.Enqueue 2.Dequeue 3.Display 4.Peek 5.Exit\n";  
 cin >> choice;  
 switch (choice) {  
 case 1: cout << "Enter value: "; cin >> val; q.enqueue(val); break;  
 case 2: q.dequeue(); break;  
 case 3: q.display(); break;  
 case 4: q.peek(); break;  
 case 5: return 0;  
 }  
 }  
}

## Q3. Interleave First Half and Second Half

#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;  
 q.push(4); q.push(7); q.push(11); q.push(20); q.push(5); q.push(9);  
  
 interleave(q);  
  
 cout << "Interleaved Queue: ";  
 while (!q.empty()) {  
 cout << q.front() << " ";  
 q.pop();  
 }  
 return 0;  
}

## Q4. First Non-Repeating Character in a String using Queue

#include <iostream>  
#include <queue>  
using namespace std;  
  
void firstNonRepeating(string str) {  
 int freq[256] = {0};  
 queue<char> q;  
  
 for (char c : str) {  
 freq[c]++;  
 q.push(c);  
 while (!q.empty() && freq[q.front()] > 1) q.pop();  
 if (q.empty()) cout << -1 << " ";  
 else cout << q.front() << " ";  
 }  
}  
  
int main() {  
 string s = "aabc";  
 cout << "Output: ";  
 firstNonRepeating(s);  
 return 0;  
}

## Q5. Stack using Queues

#include <iostream>  
#include <queue>  
using namespace std;  
  
// (a) Stack using Two Queues  
class StackTwoQ {  
 queue<int> q1, q2;  
public:  
 void push(int x) {  
 q1.push(x);  
 }  
 void pop() {  
 if (q1.empty()) return;  
 while (q1.size() > 1) {  
 q2.push(q1.front());  
 q1.pop();  
 }  
 cout << "Popped: " << q1.front() << endl;  
 q1.pop();  
 swap(q1, q2);  
 }  
};  
  
// (b) Stack using One Queue  
class StackOneQ {  
 queue<int> q;  
public:  
 void push(int x) {  
 q.push(x);  
 for (int i = 0; i < q.size() - 1; i++) {  
 q.push(q.front());  
 q.pop();  
 }  
 }  
 void pop() {  
 if (!q.empty()) {  
 cout << "Popped: " << q.front() << endl;  
 q.pop();  
 }  
 }  
};  
  
int main() {  
 StackTwoQ s1;  
 s1.push(1); s1.push(2); s1.push(3);  
 s1.pop();  
  
 StackOneQ s2;  
 s2.push(10); s2.push(20); s2.push(30);  
 s2.pop();  
 return 0;  
}