Assignment 4 by Gavisht Singh(1024030358)

1) Develop a menu driven program demonstrating the following operations on simple

Queues: enqueue(), dequeue(), isEmpty(), isFull(), display(), and peek()

Ans  
#include <iostream>

using namespace std;

#define SIZE 5

class Queue {

private:

    int arr[SIZE];

    int front;

    int rear;

public:

    Queue() {

        front = -1;

        rear = -1;

    }

    void isFull() {

       if(rear==SIZE-1){

        cout<<"Its full"<<endl;

       }

       else{

        cout<<"Its not full"<<endl;

       }

    }

    void isEmpty() {

        if(front==-1){

            cout<<"Its empty"<<endl;

        }

        else{

        cout<<"Its not empty"<<endl;

       }

    }

    void enqueue(int value) {

        if (rear==SIZE-1) {

            cout << "Queue Overflow" << endl;

        }

        if (front==-1) {

            front = 0;

        }

        rear++;

        arr[rear] = value;

        cout << value << " enqueued to queue" << endl;

    }

    void dequeue() {

        if (front==-1) {

            cout << "Queue Underflow" << endl;

        }

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

        front++;

        if (front > rear) {

            front = -1;

            rear = -1;

        }

    }

    void peek() {

        if (front==-1) {

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

        }

        cout << "Front element is: " << arr[front] << endl;

    }

    void display() {

        if (front==-1) {

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

        }

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

            cout << arr[i] << " ";

        }

        cout<<endl;

    }

};

int main() {

    Queue q;

    int choice, value;

    while (true) {

        cout << "1. Enqueue"<<endl;

        cout << "2. Dequeue"<<endl;

        cout << "3. IsEmpty"<<endl;

        cout << "4. IsFull"<<endl;

        cout << "5. Display"<<endl;

        cout << "6. Peek"<<endl;

        cout << "7. Exit"<<endl;

        cout << "Enter your choice: "<<endl;

        cin >> choice;

        switch (choice) {

            case 1:

                cout << "Enter value to enqueue: "<<endl<<endl;

                cin >> value;

                q.enqueue(value);

                break;

            case 2:

                q.dequeue();

                break;

            case 3:

                q.isEmpty();

                break;

            case 4:

                q.isFull();

                break;

            case 5:

                q.display();

                break;

            case 6:

                q.peek();

                break;

            case 7:

                cout << "Exiting program"<<endl;

                break;

            default:

                cout << "Invalid choice"<<endl;

        }

    }

*return* 0;

}

2) Develop a menu driven program demonstrating the following operations on Circular Queues:

enqueue(), dequeue(), isEmpty(), isFull(), display(), and peek().

Ans  
#include <iostream>

using namespace std;

#define SIZE 5

class CircularQueue {

private:

    int arr[SIZE];

    int front;

    int rear;

public:

    CircularQueue() {

        front = -1;

        rear = -1;

    }

    void isFull() {

        if ((front == 0 && rear == SIZE - 1) || (front == rear + 1)) {

            cout << "Its full" << endl;

        } else {

            cout << "Its not full" << endl;

        }

    }

    void isEmpty() {

        if (front == -1) {

            cout << "Its empty" << endl;

        } else {

            cout << "Its not empty" << endl;

        }

    }

    void enqueue(int value) {

        if ((front == 0 && rear == SIZE - 1) || (front == rear + 1)) {

            cout << "Queue Overflow" << endl;

*return*;

        }

        if (front == -1) {

            front = 0;

        }

        rear = (rear + 1) % SIZE;

        arr[rear] = value;

        cout << value << " enqueued to queue" << endl;

    }

    void dequeue() {

        if (front == -1) {

            cout << "Queue Underflow" << endl;

*return*;

        }

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

        if (front == rear) {

            front = -1;

            rear = -1;

        } else {

            front = (front + 1) % SIZE;

        }

    }

    void peek() {

        if (front == -1) {

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

*return*;

        }

        cout << "Front element is: " << arr[front] << endl;

    }

    void display() {

        if (front == -1) {

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

*return*;

        }

        cout << "Queue elements are: ";

        if (rear >= front) {

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

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

            }

        }

        else {

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

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

            }

            for (int i = 0; i <= rear; i++) {

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

            }

        }

        cout << endl;

    }

};

int main() {

    CircularQueue q;

    int choice, value;

    while (true) {

        cout << "1. Enqueue" << endl;

        cout << "2. Dequeue" << endl;

        cout << "3. IsEmpty" << endl;

        cout << "4. IsFull" << endl;

        cout << "5. Display" << endl;

        cout << "6. Peek" << endl;

        cout << "7. Exit" << endl;

        cout << "Enter your choice: ";

        cin >> choice;

        switch (choice) {

            case 1:

                cout << "Enter value to enqueue: ";

                cin >> value;

                q.enqueue(value);

                break;

            case 2:

                q.dequeue();

                break;

            case 3:

                q.isEmpty();

                break;

            case 4:

                q.isFull();

                break;

            case 5:

                q.display();

                break;

            case 6:

                q.peek();

                break;

            case 7:

                cout << "Exiting program"<<endl;

            default:

                cout << "Invalid choice"<<endl;

        }

    }

*return* 0;

}

3) Write a program interleave the first half of the queue with second half.

Sample I/P: 4 7 11 20 5 9 Sample O/P: 4 20 7 5 11 9

Ans  
#include <iostream>

#include <queue>

using namespace std;

void display(queue<int> q) {

    while (!q.empty()) {

        cout << q.front() << " ";

        q.pop();

    }

    cout << endl;

}

void interleave(queue<int>*&* q){

    if(q.size()<=1){

*return*;

    }

        int totalSize = q.size();

        int halfSize = totalSize / 2;

        queue<int> tempQ;

        for (int i = 0; i < halfSize; ++i) {

        tempQ.push(q.front());

        q.pop();

    }

    for (int i = 0; i < halfSize; ++i) {

        q.push(tempQ.front());

        tempQ.pop();

        q.push(q.front());

        q.pop();

    }

     if (totalSize % 2 != 0) {

        q.push(q.front());

        q.pop();

    }

}

int main(){

queue<int> q;

q.push(1);

q.push(2);

q.push(3);

q.push(4);

q.push(5);

interleave(q);

display(q);

*return* 0;

}

4) Write a program to find first non-repeating character in a string using Queue. Sample I/P: a a

b c Sample O/P: a -1 b b  
Ans

#include <iostream>

#include <queue>

using namespace std;

void find(string*&* str) {

    int count[26]={0};

    queue<char> q;

    for (char ch : str) {

        count[ch - 'a']++;

        if (count[ch - 'a'] == 1) {

            q.push(ch);

        }

        while (!q.empty() && count[q.front() - 'a'] > 1) {

            q.pop();

        }

        if (q.empty()) {

            cout << -1 << " ";

        } else {

            cout << q.front() << " ";

        }

    }

    cout << endl;

}

int main() {

    string s = "aabc";

    find(s);

*return* 0;

}

5) Write a program to implement a stack using (a) Two queues and (b) One Queue

Ans  
(Two Queues)

#include <iostream>

#include <queue>

using namespace std;

class StackWithTwoQueues {

private:

    queue<int> q1;

    queue<int> q2;

public:

    void push(int x) {

        q2.push(x);

        while (!q1.empty()) {

            q2.push(q1.front());

            q1.pop();

        }

        queue<int> temp = q1;

        q1 = q2;

        q2 = temp;

    }

    void pop() {

        if (q1.empty()) {

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

*return*;

        }

        cout << "Popping " << q1.front() << "..." << endl;

        q1.pop();

    }

    int top() {

        if (q1.empty()) {

*return* -1;

*return* q1.front();

    }

    }

    bool isEmpty() {

*return* q1.empty();

    }

};

int main() {

    StackWithTwoQueues s;

    s.push(10);

    s.push(20);

    s.push(30);

    cout << "Top element is now: " << s.top() << endl;

    s.pop();

    cout << "Top element is now: " << s.top() << endl;

    s.pop();

    cout << "Top element is now: " << s.top() << endl;

*return* 0;

}

(One queue)  
#include <iostream>

#include <queue>

using namespace std;

class StackWithOneQueue {

private:

    queue<int> q;

public:

    void push(int x) {

        int size = q.size();

        q.push(x);

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

            int frontElement = q.front();

            q.pop();

            q.push(frontElement);

        }

    }

    void pop() {

        if (isEmpty()) {

*return*;

        }

        q.pop();

    }

    int top() {

        if (isEmpty()) {

*return* -1;

        }

*return* q.front();

    }

    bool isEmpty() {

*return* q.empty();

    }

};

int main(){

StackWithOneQueue s;

s.push(1);

s.push(2);

s.push(3);

s.push(4);

cout<<"Top element is "<<s.top()<<endl;

s.pop();

cout<<"Top element is "<<s.top()<<endl;

*return* 0;

}