Assignment 4- divanshi

Q1

#include <iostream>

using namespace std;

class Queue {

int\* arr;

int front, rear, capacity;

public:

Queue(int size) {

capacity = size;

arr = new int[capacity];

front = -1;

rear = -1;

}

bool isEmpty() {

return (front == -1);

}

bool isFull() {

return (rear == capacity - 1);

}

void enqueue(int x) {

if (isFull()) {

cout << "The queue is already full!" << endl;

return;

}

if (isEmpty()) {

front = 0; // first element

}

arr[++rear] = x;

cout << x << " inserted into queue" << endl;

}

int dequeue() {

if (isEmpty()) {

cout << "The queue is empty!!" << endl;

return -1;

}

int val = arr[front];

if (front == rear) {

// only one element left

front = rear = -1;

} else {

front++;

}

cout << val << " removed from queue" << endl;

return val;

}

int peek() {

if (isEmpty()) {

cout << "Queue empty!!" << endl;

return -1;

}

return arr[front];

}

void display() {

if (isEmpty()) {

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

return;

}

cout << "The elements are: ";

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

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

}

cout << endl;

}

};

int main() {

int n;

cout << "Enter size of queue: ";

cin >> n;

Queue q(n);

int choice, val;

do {

cout << "\n--- Queue Menu ---\n";

cout << "1. Enqueue (Insert)\n";

cout << "2. Dequeue (Remove)\n";

cout << "3. Peek (Front element)\n";

cout << "4. isEmpty()\n";

cout << "5. isFull()\n";

cout << "6. Display Queue\n";

cout << "7. Exit\n";

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1:

cout << "Value to insert: ";

cin >> val;

q.enqueue(val);

break;

case 2:

q.dequeue();

break;

case 3:

val = q.peek();

if (val != -1) cout << "Front element is: " << val << endl;

break;

case 4:

if (q.isEmpty())

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

else

cout << "Queue is not empty" << endl;

break;

case 5:

if (q.isFull())

cout << "The queue is full" << endl;

else

cout << "The queue is not full" << endl;

break;

case 6:

q.display();

break;

case 7:

cout << "Exit" << endl;

break;

default:

cout << "Invalid choice! Try again." << endl;

}

} while (choice != 7);

return 0;

}

Q2)

#include <iostream>

using namespace std;

class CircularQueue {

int\* arr;

int front, rear, capacity;

public:

CircularQueue(int size) {

capacity = size;

arr = new int[capacity];

front = -1;

rear = -1;

}

bool isEmpty() {

return (front == -1);

}

bool isFull() {

return ((rear + 1) % capacity == front);

}

void enqueue(int x) {

if (isFull()) {

cout << "Queue Overflow! Cannot insert " << x << endl;

return;

}

if (isEmpty()) {

front = rear = 0;

} else {

rear = (rear + 1) % capacity;

}

arr[rear] = x;

cout << x << " inserted into the circular queue." << endl;

}

int dequeue() {

if (isEmpty()) {

cout << "Queue Underflow! Nothing to delete." << endl;

return -1;

}

int val = arr[front];

if (front == rear) {

front = rear = -1;

} else {

front = (front + 1) % capacity;

}

cout << val << " removed from the circular queue." << endl;

return val;

}

int peek() {

if (isEmpty()) {

cout << "Queue is empty. Nothing at front." << endl;

return -1;

}

return arr[front];

}

void display() {

if (isEmpty()) {

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

return;

}

cout << "Queue elements: ";

int i = front;

while (true) {

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

if (i == rear) break;

i = (i + 1) % capacity;

}

cout << endl;

}

};

int main() {

int n;

cout << "Enter size of circular queue: ";

cin >> n;

CircularQueue q(n);

int choice, val;

do {

cout << "\n---- Circular Queue Menu ----\n";

cout << "1. Enqueue (Insert)\n";

cout << "2. Dequeue (Remove)\n";

cout << "3. Peek (Front element)\n";

cout << "4. isEmpty()\n";

cout << "5. isFull()\n";

cout << "6. Display Queue\n";

cout << "7. Exit\n";

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1:

cout << "Enter value to insert: ";

cin >> val;

q.enqueue(val);

break;

case 2:

q.dequeue();

break;

case 3:

val = q.peek();

if (val != -1) cout << "Front element is: " << val << endl;

break;

case 4:

if (q.isEmpty()) cout << "Queue is empty." << endl;

else cout << "Queue is not empty." << endl;

break;

case 5:

if (q.isFull()) cout << "Queue is full." << endl;

else cout << "Queue is not full." << endl;

break;

case 6:

q.display();

break;

case 7:

cout << "Exiting program..." << endl;

break;

default:

cout << "Invalid choice! Please try again." << endl;

}

} while (choice != 7);

return 0;

}

Q3

#include <iostream>

#include <queue>

using namespace std;

int main() {

queue<int> q;

int arr[] = {4, 7, 11, 20, 5, 9};

int n = 6;

for (int i = 0; i < n; i++) q.push(arr[i]);

int half = n / 2;

queue<int> firstHalf;

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

firstHalf.push(q.front());

q.pop();

}

while (!firstHalf.empty()) {

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

firstHalf.pop();

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

q.pop();

}

return 0;

}

Q4)

#include <iostream>

#include <queue>

using namespace std;

int main() {

string s = "aabc";

queue<char> q;

int freq[26] = {0};

for (char ch : s) {

q.push(ch);

freq[ch - 'a']++;

while (!q.empty() && freq[q.front() - 'a'] > 1) q.pop();

if (q.empty()) cout << -1 << " ";

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

}

}

Q5\_a)

#include <iostream>

#include <queue>

using namespace std;

class StackTwoQ {

queue<int> q1, q2;

public:

void push(int x) {

q2.push(x);

while (!q1.empty()) {

q2.push(q1.front());

q1.pop();

}

swap(q1, q2);

}

void pop() {

if (!q1.empty()) q1.pop();

else cout << "Stack is empty\n";

}

int top() {

if (!q1.empty()) return q1.front();

return -1;

}

bool empty() {

return q1.empty();

}

};

int main() {

StackTwoQ st;

st.push(10);

st.push(20);

st.push(30);

cout << st.top() << endl; // 30

st.pop();

cout << st.top() << endl; // 20

return 0;

}

Q5\_b)

#include <iostream>

#include <queue>

using namespace std;

class StackOneQ {

queue<int> q;

public:

void push(int x) {

int size = q.size();

q.push(x);

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

q.push(q.front());

q.pop();

}

}

void pop() {

if (!q.empty()) q.pop();

else cout << "Stack is empty\n";

}

int top() {

if (!q.empty()) return q.front();

return -1;

}

bool empty() {

return q.empty();

}

};

int main() {

StackOneQ st;

st.push(5);

st.push(15);

st.push(25);

cout << st.top() << endl; // 25

st.pop();

cout << st.top() << endl; // 15

return 0;

}