DSA

Lab assignment-4

Name:- Harshvardhan singh Naruka

Roll no:- 1024030341

(Q1)

#include <iostream>

using namespace std;

#define MAX 5

class Queue {

int arr[MAX];

int front, rear;

public:

Queue() {

front = -1;

rear = -1;

}

bool isEmpty() {

return (front == -1);

}

bool isFull() {

return (rear == MAX - 1);

}

void enqueue(int x) {

if (isFull()) {

cout << "Queue is Full!\n";

return;

}

if (front == -1) front = 0; // first element

arr[++rear] = x;

cout << x << " enqueued.\n";

}

void dequeue() {

if (isEmpty()) {

cout << "Queue is Empty!\n";

return;

}

cout << arr[front] << " dequeued.\n";

if (front == rear) {

front = rear = -1; // reset queue

} else {

front++;

}

}

void peek() {

if (isEmpty()) {

cout << "Queue is Empty!\n";

} else {

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

}

}

void display() {

if (isEmpty()) {

cout << "Queue is Empty!\n";

return;

}

cout << "Queue elements: ";

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

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

}

cout << endl;

}

};

int main() {

Queue q;

int choice, value;

do {

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

cout << "1. Enqueue\n2. Dequeue\n3. Peek\n4. Display\n5. Exit\n";

cout << "Enter choice: ";

cin >> choice;

switch (choice) {

case 1:

cout << "Enter value: ";

cin >> value;

q.enqueue(value);

break;

case 2:

q.dequeue();

break;

case 3:

q.peek();

break;

case 4:

q.display();

break;

case 5:

cout << "Exiting...\n";

break;

default:

cout << "Invalid choice!\n";

}

} while (choice != 5);

return 0;

}

(Q2)

#include <iostream>

using namespace std;

#define MAX 5

class CircularQueue {

int arr[MAX];

int front, rear;

public:

CircularQueue() {

front = -1;

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 (isEmpty()) {

front = rear = 0;

} else {

rear = (rear + 1) % MAX;

}

arr[rear] = x;

cout << x << " enqueued.\n";

}

void dequeue() {

if (isEmpty()) {

cout << "Queue is Empty!\n";

return;

}

cout << arr[front] << " dequeued.\n";

if (front == rear) {

// only one element

front = rear = -1;

} else {

front = (front + 1) % MAX;

}

}

void peek() {

if (isEmpty()) {

cout << "Queue is Empty!\n";

} else {

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

}

}

void display() {

if (isEmpty()) {

cout << "Queue is Empty!\n";

return;

}

cout << "Queue elements: ";

int i = front;

while (true) {

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

if (i == rear) break;

i = (i + 1) % MAX;

}

cout << endl;

}

};

int main() {

CircularQueue cq;

int choice, value;

do {

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

cout << "1. Enqueue\n2. Dequeue\n3. Peek\n4. Display\n5. Exit\n";

cout << "Enter choice: ";

cin >> choice;

switch (choice) {

case 1:

cout << "Enter value: ";

cin >> value;

cq.enqueue(value);

break;

case 2:

cq.dequeue();

break;

case 3:

cq.peek();

break;

case 4:

cq.display();

break;

case 5:

cout << "Exiting...\n";

break;

default:

cout << "Invalid choice!\n";

}

} while (choice != 5);

return 0;

}

(Q3)

#include <iostream>

#include <queue>

using namespace std;

void interleaveQueue(queue<int>& q) {

if (q.size() % 2 != 0) {

cout << "Queue has odd number of elements, cannot interleave!\n";

return;

}

int halfSize = q.size() / 2;

queue<int> firstHalf;

// Step 1: Push first half into another queue

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

firstHalf.push(q.front());

q.pop();

}

// Step 2: Interleave the two halves

while (!firstHalf.empty()) {

q.push(firstHalf.front());

firstHalf.pop();

q.push(q.front());

q.pop();

}

}

void display(queue<int> q) {

while (!q.empty()) {

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

q.pop();

}

cout << endl;

}

int main() {

queue<int> q;

int n, val;

cout << "Enter number of elements (even): ";

cin >> n;

cout << "Enter " << n << " elements:\n";

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

cin >> val;

q.push(val);

}

cout << "Original Queue: ";

display(q);

interleaveQueue(q);

cout << "Interleaved Queue: ";

display(q);

return 0;

}

(Q4)

#include <iostream>

#include <queue>

#include <unordered\_map>

using namespace std;

void firstNonRepeating(string s) {

queue<char> q;

unordered\_map<char, int> freq;

for (char ch : s) {

freq[ch]++;

q.push(ch);

while (!q.empty() && freq[q.front()] > 1) {

q.pop();

}

if (q.empty())

cout << -1 << " ";

else

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

}

cout << endl;

}

int main() {

string s;

cout << "Enter string: ";

cin >> s;

cout << "First non-repeating characters: ";

firstNonRepeating(s);

return 0;

}

(Q5)

(A)

#include <iostream>

#include <queue>

using namespace std;

class StackUsingTwoQueues {

queue<int> q1, q2;

public:

void push(int x) {

q2.push(x);

while (!q1.empty()) {

q2.push(q1.front());

q1.pop();

}

swap(q1, q2);

cout << x << " pushed.\n";

}

void pop() {

if (q1.empty()) {

cout << "Stack is Empty!\n";

return;

}

cout << q1.front() << " popped.\n";

q1.pop();

}

void top() {

if (q1.empty()) {

cout << "Stack is Empty!\n";

return;

}

cout << "Top element: " << q1.front() << endl;

}

void display() {

if (q1.empty()) {

cout << "Stack is Empty!\n";

return;

}

cout << "Stack elements: ";

queue<int> temp = q1;

while (!temp.empty()) {

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

temp.pop();

}

cout << endl;

}

};

int main() {

StackUsingTwoQueues st;

int choice, value;

do {

cout << "\n--- Stack (Two Queues) Menu ---\n";

cout << "1. Push\n2. Pop\n3. Top\n4. Display\n5. Exit\n";

cout << "Enter choice: ";

cin >> choice;

switch (choice) {

case 1:

cout << "Enter value: ";

cin >> value;

st.push(value);

break;

case 2:

st.pop();

break;

case 3:

st.top();

break;

case 4:

st.display();

break;

case 5:

cout << "Exiting...\n";

break;

default:

cout << "Invalid choice!\n";

}

} while (choice != 5);

return 0;

}

(B)

#include <iostream>

#include <queue>

using namespace std;

class StackUsingOneQueue {

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();

}

cout << x << " pushed.\n";

}

void pop() {

if (q.empty()) {

cout << "Stack is Empty!\n";

return;

}

cout << q.front() << " popped.\n";

q.pop();

}

void top() {

if (q.empty()) {

cout << "Stack is Empty!\n";

return;

}

cout << "Top element: " << q.front() << endl;

}

void display() {

if (q.empty()) {

cout << "Stack is Empty!\n";

return;

}

cout << "Stack elements: ";

queue<int> temp = q;

while (!temp.empty()) {

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

temp.pop();

}

cout << endl;

}

};

int main() {

StackUsingOneQueue st;

int choice, value;

do {

cout << "\n--- Stack (One Queue) Menu ---\n";

cout << "1. Push\n2. Pop\n3. Top\n4. Display\n5. Exit\n";

cout << "Enter choice: ";

cin >> choice;

switch (choice) {

case 1:

cout << "Enter value: ";

cin >> value;

st.push(value);

break;

case 2:

st.pop();

break;

case 3:

st.top();

break;

case 4:

st.display();

break;

case 5:

cout << "Exiting...\n";

break;

default:

cout << "Invalid choice!\n";

}

} while (choice != 5);

return 0;

}