*1. Develop a menu driven program demonstrating the following operations on simple Queues: enqueue(), dequeue(), isEmpty(), isFull(), display(), and peek().*

#include <iostream>

using namespace std;

#define SIZE 100

class Queue {

int arr[SIZE];

int front, rear;

public:

Queue() {

front = -1;

rear = -1;

}

void enqueue(int value) {

if (isFull()) {

cout << "Queue is full\n";

return;

}

if (isEmpty()) {

front = 0;

}

rear++;

arr[rear] = value;

}

void dequeue() {

if (isEmpty()) {

cout << "Queue is empty\n";

return;

}

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

front++;

if (front > rear) {

front = rear = -1;

}

}

bool isEmpty() {

return front == -1;

}

bool isFull() {

return rear == SIZE - 1;

}

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;

while (true) {

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

cout << "1. Enqueue\n";

cout << "2. Dequeue\n";

cout << "3. Peek\n";

cout << "4. Check if Empty\n";

cout << "5. Check if Full\n";

cout << "6. Display\n";

cout << "7. Exit\n";

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.peek();

break;

case 4:

if (q.isEmpty())

cout << "Queue is empty\n";

else

cout << "Queue is not empty\n";

break;

case 5:

if (q.isFull())

cout << "Queue is full\n";

else

cout << "Queue is not full\n";

break;

case 6:

q.display();

break;

case 7:

return 0;

default:

cout << "Invalid choice\n";

}

}

}

*2. Develop a menu driven program demonstrating the following operations on Circular Queues: enqueue(), dequeue(), isEmpty(), isFull(), display(), and peek().*

#include <iostream>

using namespace std;

#define SIZE 5

class CircularQueue {

int arr[SIZE];

int front, rear;

public:

CircularQueue() {

front = -1;

rear = -1;

}

bool isEmpty() {

return front == -1;

}

bool isFull() {

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

}

void enqueue(int value) {

if (isFull()) {

cout << "Queue is full\n";

return;

}

if (isEmpty()) {

front = rear = 0;

} else {

rear = (rear + 1) % SIZE;

}

arr[rear] = value;

}

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) % SIZE;

}

}

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) % SIZE;

}

cout << endl;

}

};

int main() {

CircularQueue q;

int choice, value;

while (true) {

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

cout << "1. Enqueue\n";

cout << "2. Dequeue\n";

cout << "3. Peek\n";

cout << "4. Check if Empty\n";

cout << "5. Check if Full\n";

cout << "6. Display\n";

cout << "7. Exit\n";

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.peek();

break;

case 4:

if (q.isEmpty())

cout << "Queue is empty\n";

else

cout << "Queue is not empty\n";

break;

case 5:

if (q.isFull())

cout << "Queue is full\n";

else

cout << "Queue is not full\n";

break;

case 6:

q.display();

break;

case 7:

return 0;

default:

cout << "Invalid choice\n";

}

}

}

*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*

#include <iostream>

#include <queue>

#include <stack>

using namespace std;

void interleaveQueue(queue<int>& q) {

int n = q.size();

if (n % 2 != 0) {

cout << "Queue length must be even\n";

return;

}

queue<int> firstHalf;

for (int i = 0; i < n / 2; 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;

int x;

cout << "Enter even number of elements (end with -1): ";

while (cin >> x && x != -1){

q.push(x);

}

interleaveQueue(q);

cout << "Interleaved Queue: ";

while (!q.empty()) {

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

q.pop();

}

cout << endl;

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*

#include <iostream>

#include <queue>

#include <unordered\_map>

using namespace std;

int main() {

string str;

getline(cin, str);

queue<char> q;

unordered\_map<char, int> freq;

for (char ch : str) {

if (ch == ' ') continue;

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;

return 0;

}

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

*A) Using two queues*

#include <iostream>

#include <queue>

using namespace std;

class StackTwoQueues {

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()) {

cout << "Stack is empty\n";

return;

}

cout << "Popped: " << q1.front() << endl;

q1.pop();

}

int top() {

if (q1.empty()) {

cout << "Stack is empty\n";

return -1;

}

return q1.front();

}

bool empty() {

return q1.empty();

}

};

int main() {

StackTwoQueues s;

s.push(10);

s.push(20);

s.push(30);

cout << s.top() << endl;

s.pop();

cout << s.top() << endl;

s.pop();

cout << s.top() << endl;

s.pop();

s.pop();

return 0;

}

*B) Using one queue*

#include <iostream>

#include <queue>

using namespace std;

class StackOneQueue {

queue<int> q;

public:

void push(int x) {

q.push(x);

int sz = q.size();

for (int i = 0; i < sz - 1; i++) {

q.push(q.front());

q.pop();

}

}

void pop() {

if (q.empty()) {

cout << "Stack is empty\n";

return;

}

cout << "Popped: " << q.front() << endl;

q.pop();

}

int top() {

if (q.empty()) {

cout << "Stack is empty\n";

return -1;

}

return q.front();

}

bool empty() {

return q.empty();

}

};

int main() {

StackOneQueue s;

s.push(10);

s.push(20);

s.push(30);

cout << s.top() << endl;

s.pop();

cout << s.top() << endl;

s.pop();

cout << s.top() << endl;

s.pop();

s.pop();

return 0;

}