Name: Aryan Juneja roll no. :- 1024030920 Batch :- 2C64

Q1

#include <iostream>

using namespace std;

const int MAX\_SIZE = 100;

class SimpleQueue

{

private:

int arr[MAX\_SIZE];

int front;

int rear;

public:

SimpleQueue()

{

front = -1;

rear = -1;

}

bool isEmpty()

{

return (front == -1);

}

bool isFull()

{

return (rear == MAX\_SIZE - 1);

}

void enqueue(int value)

{

if (isFull())

{

cout << "Queue is full. Cannot enqueue.\n";

return;

}

if (isEmpty())

{

front = 0;

}

rear++;

arr[rear] = value;

cout << "Enqueued: " << value << "\n";

}

void dequeue()

{

if (isEmpty())

{

cout << "Queue is empty. Cannot dequeue.\n";

return;

}

int dequeuedValue = arr[front];

cout << "Dequeued: " << dequeuedValue << "\n";

if (front == rear)

{

// Last element is dequeued, reset queue

front = -1;

rear = -1;

}

else

{

front++;

}

}

void peek()

{

if (isEmpty())

{

cout << "Queue is empty. No element to peek.\n";

return;

}

cout << "Element at the front is: " << arr[front] << "\n";

}

void display()

{

if (isEmpty())

{

cout << "Queue is empty.\n";

return;

}

cout << "Queue elements are: ";

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

{

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

}

cout << "\n";

}

};

void showMenu()

{

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

cout << "1. Enqueue\n";

cout << "2. Dequeue\n";

cout << "3. Peek\n";

cout << "4. Display\n";

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

cout << "6. Check if Full\n";

cout << "7. Exit\n";

cout << "-------------------------\n";

cout << "Enter your choice: ";

}

int main()

{

SimpleQueue queue;

int choice = 0;

int value;

while (choice != 7)

{

showMenu();

cin >> choice;

switch (choice)

{

case 1:

cout << "Enter value to enqueue: ";

cin >> value;

queue.enqueue(value);

break;

case 2:

queue.dequeue();

break;

case 3:

queue.peek();

break;

case 4:

queue.display();

break;

case 5:

if (queue.isEmpty())

{

cout << "Queue is empty.\n";

}

else

{

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

}

break;

case 6:

if (queue.isFull())

{

cout << "Queue is full.\n";

}

else

{

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

}

break;

case 7:

cout << "Exiting program.\n";

break;

default:

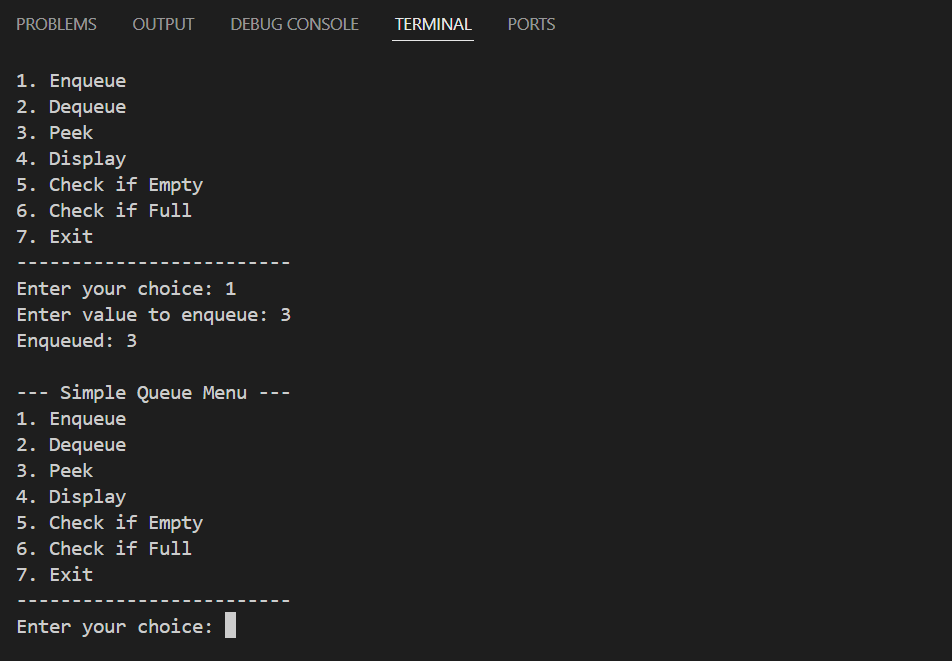
cout << "Invalid choice. Please try again.\n";

}

}

return 0;

}



Q2

#include <iostream>

using namespace std;

class CircularQueue {

private:

int\* arr;

int front;

int rear;

int size;

int count;

public:

CircularQueue(int s) {

size = s;

arr = new int[size];

front = -1;

rear = -1;

count = 0;

}

~CircularQueue() {

delete[] arr;

}

bool isFull() {

return (count == size);

}

bool isEmpty() {

return (count == 0);

}

void enqueue(int value) {

if (isFull()) {

cout << "Queue is full. Cannot enqueue.\n";

return;

}

if (isEmpty()) {

front = 0;

rear = 0;

} else {

rear = (rear + 1) % size;

}

arr[rear] = value;

count++;

cout << "Enqueued: " << value << "\n";

}

void dequeue() {

if (isEmpty()) {

cout << "Queue is empty. Cannot dequeue.\n";

return;

}

int dequeuedValue = arr[front];

cout << "Dequeued: " << dequeuedValue << "\n";

if (front == rear) {

front = -1;

rear = -1;

} else {

front = (front + 1) % size;

}

count--;

}

void peek() {

if (isEmpty()) {

cout << "Queue is empty. No element to peek.\n";

return;

}

cout << "Element at the front is: " << arr[front] << "\n";

}

void display() {

if (isEmpty()) {

cout << "Queue is empty.\n";

return;

}

cout << "Queue elements are: ";

int i = front;

while (i != rear) {

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

i = (i + 1) % size;

}

cout << arr[rear] << "\n";

}

};

void showMenu() {

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

cout << "1. Enqueue\n";

cout << "2. Dequeue\n";

cout << "3. Peek\n";

cout << "4. Display\n";

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

cout << "6. Check if Full\n";

cout << "7. Exit\n";

cout << "---------------------------\n";

cout << "Enter your choice: ";

}

int main() {

int size;

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

cin >> size;

CircularQueue cq(size);

int choice = 0;

int value;

while (choice != 7) {

showMenu();

cin >> choice;

switch (choice) {

case 1:

cout << "Enter value to enqueue: ";

cin >> value;

cq.enqueue(value);

break;

case 2:

cq.dequeue();

break;

case 3:

cq.peek();

break;

case 4:

cq.display();

break;

case 5:

if (cq.isEmpty()) {

cout << "Queue is empty.\n";

} else {

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

}

break;

case 6:

if (cq.isFull()) {

cout << "Queue is full.\n";

} else {

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

}

break;

case 7:

cout << "Exiting program.\n";

break;

default:

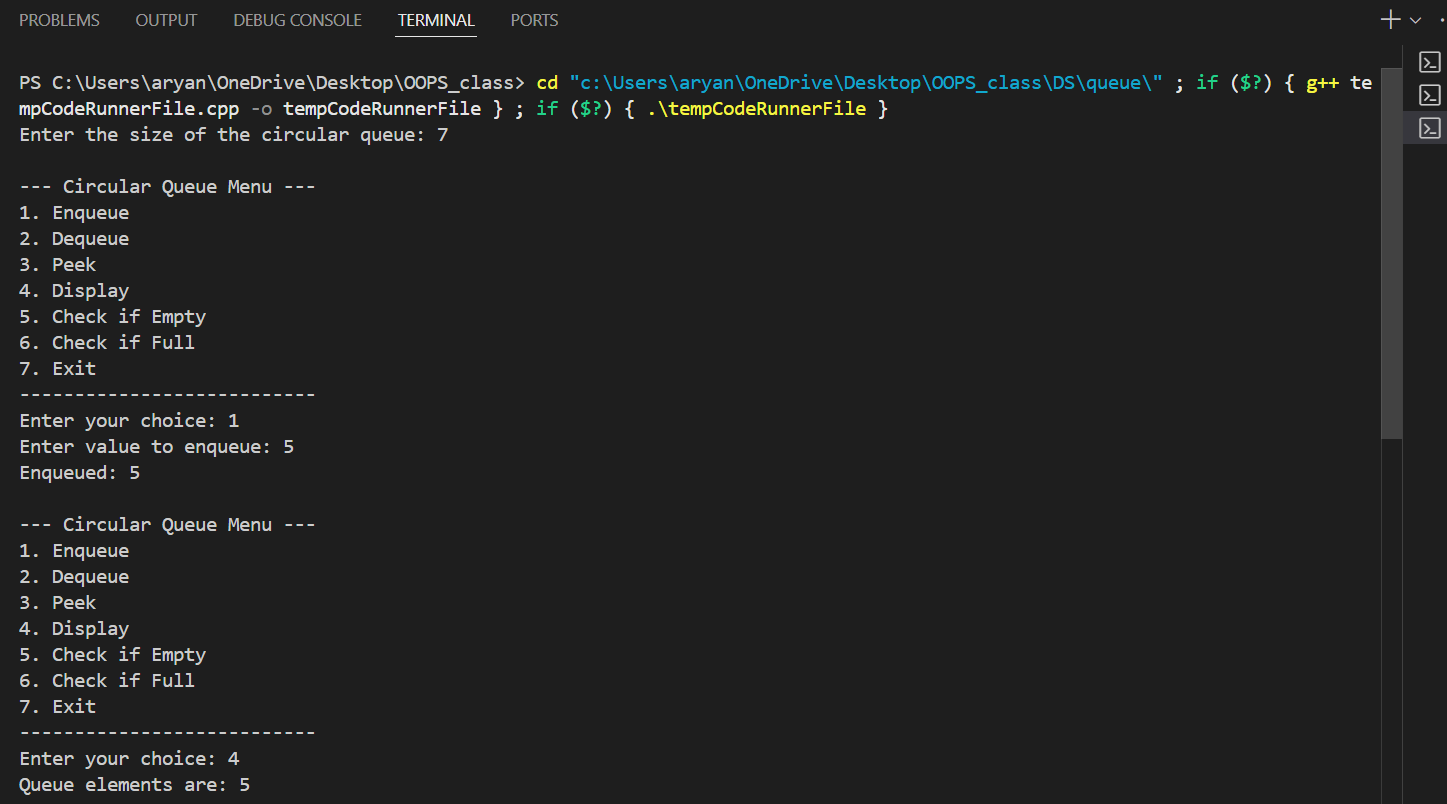
cout << "Invalid choice. Please try again.\n";

}

}

return 0;

}



Q3

#include <iostream>

#include <queue>

#include <stack>

using namespace std;

void interleaveQueue(queue<int>& q) {

if (q.empty()) return;

int halfSize = q.size() / 2;

stack<int> s;

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

s.push(q.front());

q.pop();

}

while (!s.empty()) {

q.push(s.top());

s.pop();

}

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

q.push(q.front());

q.pop();

}

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

s.push(q.front());

q.pop();

}

while (!s.empty()) {

q.push(s.top());

s.pop();

q.push(q.front());

q.pop();

}

}

int main() {

queue<int> q;

int n, x;

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

cin >> n;

if (n % 2 != 0) {

cout << "Number of elements should be even.\n";

return 1;

}

cout << "Enter elements:\n";

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

cin >> x;

q.push(x);

}

interleaveQueue(q);

cout << "Interleaved queue: ";

while (!q.empty()) {

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

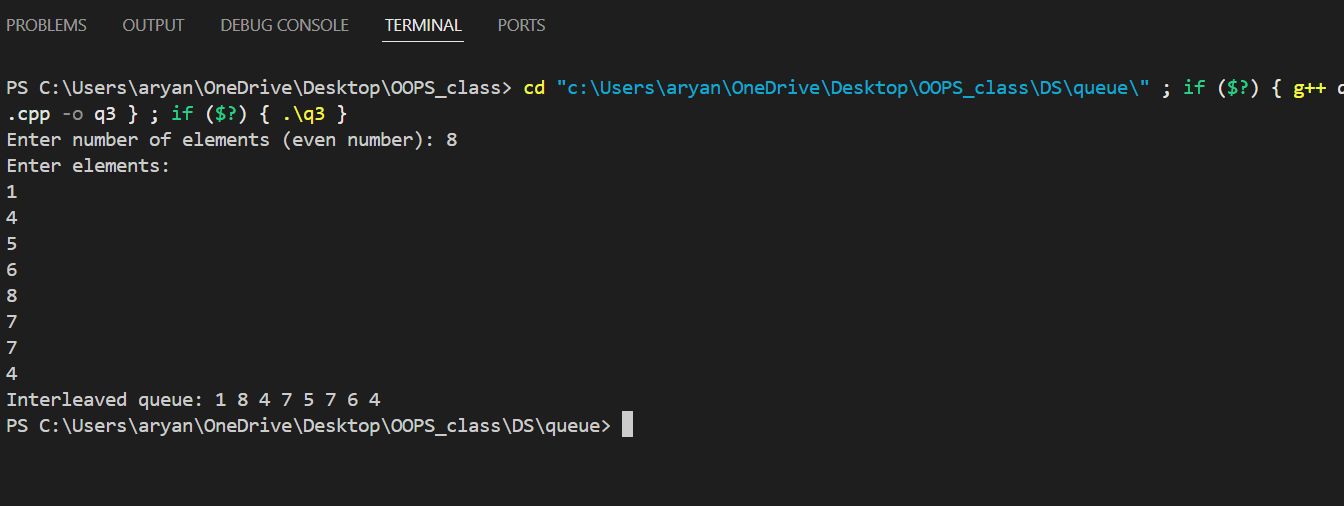
q.pop();

}

cout << "\n";

return 0;

}



Q4

#include <iostream>

#include <queue>

#include <string>

using namespace std;

string Conversion(string &ch)

{

queue<char> qu;

int n = ch.length();

int i;

string st = "";

for (char c : ch)

{

if (qu.empty())

{

qu.push(c);

st += c;

}

else if (c == qu.front())

{

qu.pop();

st += "-1";

}

else

{

st += qu.front();

}

}

return st;

}

int main()

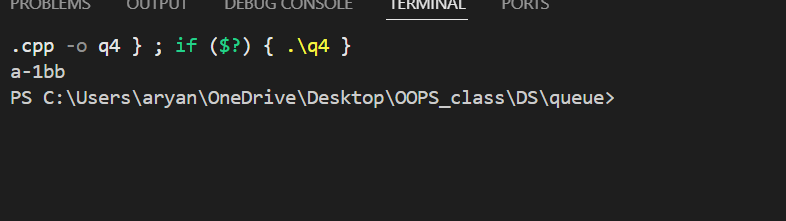
{

string ch = "aabc";

ch = Conversion(ch);

cout << ch;

}



Q5

A

#include <bits/stdc++.h>

using namespace std;

class StackWithTwoQueues

{

private:

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. Cannot pop." << endl;

return;

}

q1.pop();

}

int top()

{

if (q1.empty())

{

cout << "Stack is empty. No top element." << endl;

return -1;

}

return q1.front();

}

bool empty()

{

return q1.empty();

}

};

void runTwoQueueExample()

{

StackWithTwoQueues s;

s.push(1);

s.push(2);

s.push(3);

cout << "Top element is: " << s.top() << endl; // Output: 3

s.pop();

cout << "Top element after pop is: " << s.top() << endl; // Output: 2

}

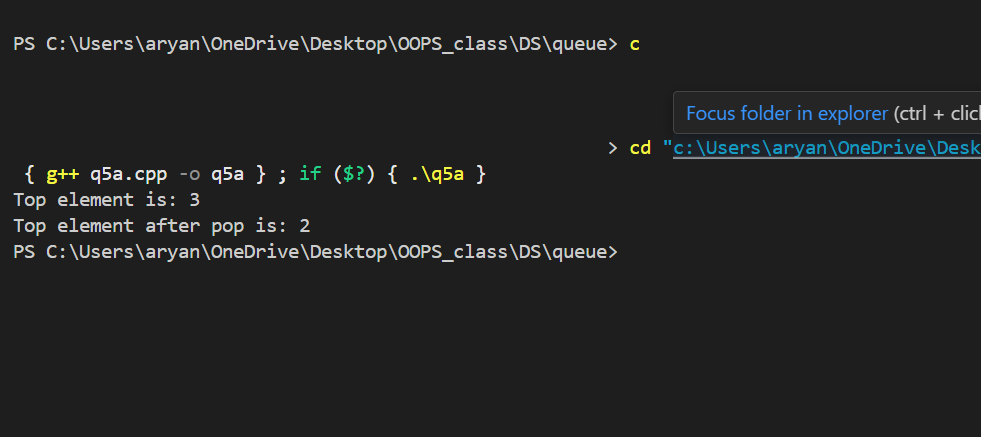
int main()

{

runTwoQueueExample();

return 0;

}



Q5

B

#include <iostream>

#include <queue>

using namespace std;

class StackWithOneQueue

{

private:

queue<int> q;

public:

void push(int x)

{

int n = q.size();

q.push(x);

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

{

q.push(q.front());

q.pop();

}

}

void pop()

{

if (q.empty())

{

cout << "Stack is empty. Cannot pop." << endl;

return;

}

q.pop();

}

int top()

{

if (q.empty())

{

cout << "Stack is empty. No top element." << endl;

return -1;

}

return q.front();

}

bool empty()

{

return q.empty();

}

};

void runOneQueueExample()

{

StackWithOneQueue s;

s.push(10);

s.push(20);

s.push(30);

cout << "Top element is: " << s.top() << endl; // Output: 30

s.pop();

cout << "Top element after pop is: " << s.top() << endl; // Output: 20

}

int main()

{

runOneQueueExample();

return 0;

}

