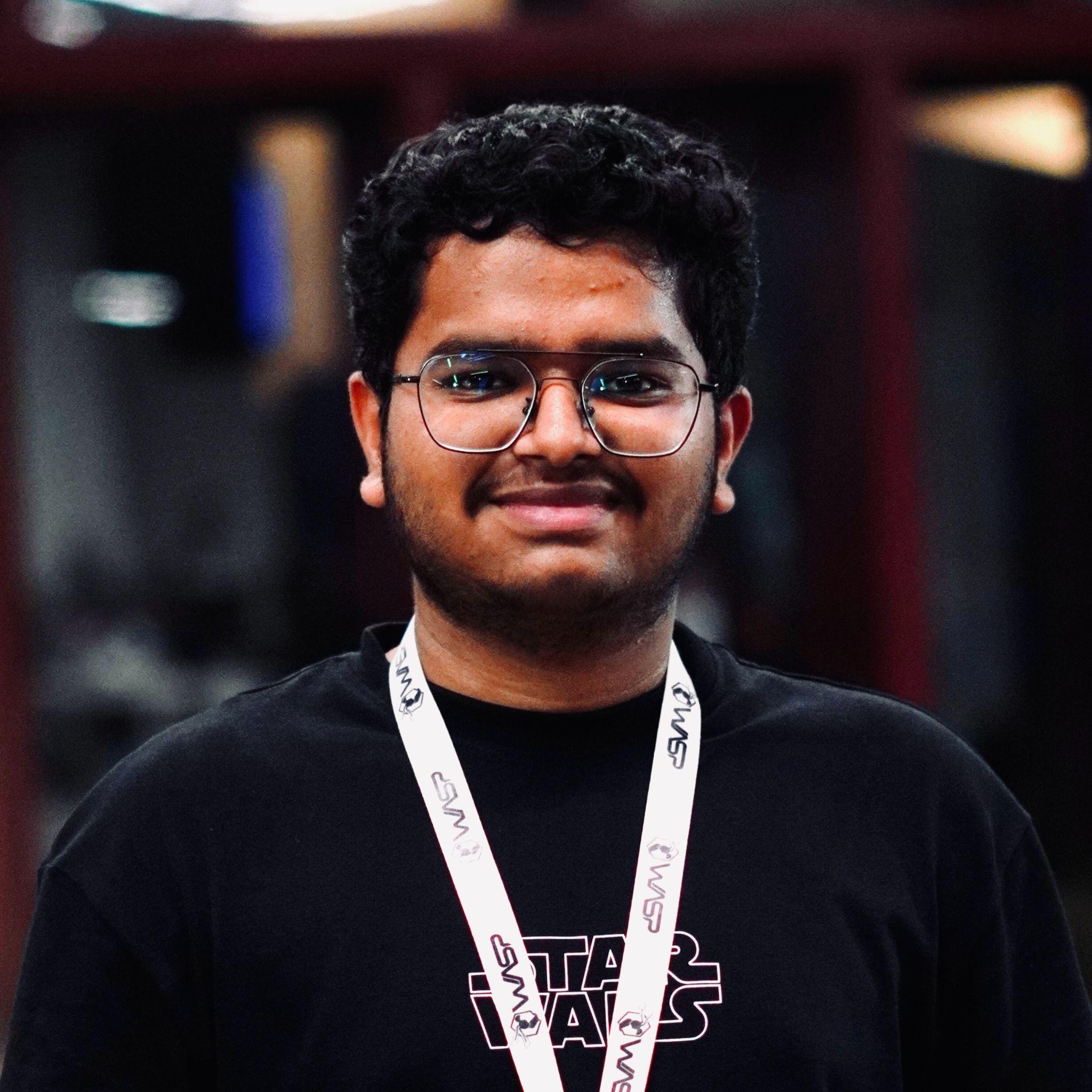
Name: Dhruv Mittal

Roll no: 1024030364

Subgroup: 2C24



**Q-1 :**

/\*

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 ;

class Queue {

public:

int \*arr ;

int front ;

int rear ;

int size ;

Queue(int size){

this->size = size ;

arr = new int[size] ;

front = -1 ;

rear = -1 ;

}

void push(int x){ // push() -> enqueue()

if(rear == size){

cout << "Queue is FULL !" << endl ;

}

else{

arr[rear] = x ;

rear ++ ;

}

}

void pop(){ // pop() -> dequeue()

if(front == -1 && rear == -1){

cout << "Queue is EMPTY !" << endl ;

}

else{

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

if(front == rear){

front = rear = -1 ;

}

else{

front++ ;

}

}

}

int peek(){ // peek()

if(front == rear){

return -1 ; // empty queue

}

else{

return arr[front] ;

}

}

bool isEmpty(){

return (front == -1 && rear == -1);

}

bool isFull(){

return (rear == size - 1);

}

void display(){

if(isEmpty()){

cout << "Queue is EMPTY !" ;

}

else{

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

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

}

cout << endl ;

}

}

} ;

int main(){

Queue q(5) ;

q.push(10);

q.push(20);

q.push(30);

cout << "Front " << q.peek() << endl ;

q.pop() ;

q.pop() ;

cout << "Front " << q.peek() << endl ;

q.push(40);

q.push(4);

q.push(7);

q.display() ;

return 0 ;

}

**Q-2 :**

/\*

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 ;

class Circular\_Queue {

int \*arr ;

int curr\_size ;

int cap ;

int f , r ;

public:

Circular\_Queue(int size){

this->cap = size ;

arr = new int[size] ;

curr\_size = 0 ;

f = 0 ;

f = -1 ;

}

void push(int x){ // push -> enqueue

if(isFull()){

cout << "CQ is full !" << endl ;

return ;

}

r = (r+1) % cap ;

arr[r] = x ;

curr\_size ++ ;

}

void pop(){ // pop -> dequeue

if(isEmpty()){

cout << "CQ is EMPTY !" << endl ;

return ;

}

f = (f+1) % cap ;

curr\_size -- ;

}

bool isEmpty(){

return (curr\_size == 0) ;

}

bool isFull(){

return (curr\_size == cap) ;

}

void display(){

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

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

}

cout << endl ;

}

int front(){ // front -> peek

if(isEmpty()){

cout << "CQ is EMPTY !" << endl ;

return -1 ;

}

return arr[f] ;

}

} ;

int main(){

Circular\_Queue c(3) ;

c.push(5) ;

c.push(3) ;

c.push(1) ;

c.pop() ;

c.push(4) ;

c.display() ;

return 0 ;

}

**Q-3 :**

/\*

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>

using namespace std;

void interleaveQueue(queue<int> &q) {

int n = q.size();

if (n % 2 != 0) {

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

return;

}

int half = n / 2;

queue<int> firstHalf;

// Store first half in another queue

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

firstHalf.push(q.front());

q.pop();

}

// Now interleave

while (!firstHalf.empty()) {

q.push(firstHalf.front());

firstHalf.pop();

q.push(q.front());

q.pop();

}

}

int main() {

queue<int> q;

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

int n = sizeof(arr) / sizeof(arr[0]);

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

q.push(arr[i]);

interleaveQueue(q);

// Print output

while (!q.empty()) {

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

q.pop();

}

cout << endl;

return 0;

}

**Q-4 :**

**Q-5 :**

/\*

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

\*/

#include <iostream>

#include <queue>

using namespace std ;

class StackTwoQueues {

public:

queue<int> q1, q2;

void push(int x) {

while (!q1.empty()) {

q2.push(q1.front());

q1.pop();

}

q1.push(x);

while (!q2.empty()) {

q1.push(q2.front());

q2.pop();

}

}

int pop() {

if (q1.empty()) return -1;

int ans = q1.front();

q1.pop();

return ans;

}

int top() {

if (q1.empty()) return -1;

return q1.front();

}

bool empty() {

return q1.empty();

}

};

class StackOneQueue {

public:

queue<int> q;

void push(int x) {

int size = q.size();

q.push(x);

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

q.push(q.front());

q.pop();

}

}

int pop() {

if (q.empty()) return -1;

int ans = q.front();

q.pop();

return ans;

}

int top() {

if (q.empty()) return -1;

return q.front();

}

bool empty() {

return q.empty();

}

};

int main() {

cout << "Stack using Two Queues:" << endl;

StackTwoQueues s1;

s1.push(10);

s1.push(20);

s1.push(30);

cout << "Top: " << s1.top() << endl;

cout << "Pop: " << s1.pop() << endl;

cout << "Top: " << s1.top() << endl;

cout << "\nStack using One Queue:" << endl;

StackOneQueue s2;

s2.push(5);

s2.push(15);

s2.push(25);

cout << "Top: " << s2.top() << endl;

cout << "Pop: " << s2.pop() << endl;

cout << "Top: " << s2.top() << endl;

return 0;

}