NAME – Ridhima goel BATCH – 2C32

ROLL NUMBER: 1024030466 ASSIGNEMENT – 4(QUEUE)

Ques1. Develop a menu driven program demonstrating the following operations on simple Queues: enqueue(), dequeue(), isEmpty(), isFull(), display(), and peek ().

Soln: #include <iostream> using namespace std; #define SIZE 5

class Queue { private:

int arr[SIZE];

int front, rear;

public:

Queue() {

front = -1;

rear = -1;

}

bool isEmpty() {

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

}

bool isFull() {

return (rear == SIZE - 1);

}

void enqueue(int value) {

if (isFull()) {

cout << "Queue Overflow! Cannot insert " << value << endl; return;

}

if (isEmpty()) {

front = rear = 0;

} else {

rear++;

}

arr[rear] = value;

cout << value << " enqueued into the queue." << endl;

}

void dequeue() { if (isEmpty()) {

cout << "Queue Underflow! Cannot dequeue." << endl; return;

}

cout << arr[front] << " dequeued from the queue." << endl; if (front == rear) {

front = rear = -1;

} else {

front++;

}

}

void peek() {

if (isEmpty()) {

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

} else {

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

}

}

void display() { if (isEmpty()) {

cout << "Queue is Empty!" << endl; 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 ========" << endl; cout << "1. Enqueue" << endl;

cout << "2. Dequeue" << endl; cout << "3. Peek" << endl;

cout << "4. Display" << endl;

cout << "5. Check if Empty" << endl; cout << "6. Check if Full" << endl;

cout << "0. Exit" << endl;

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:

q.display(); break;

case 5:

if (q.isEmpty())

cout << "Queue is Empty." << endl; else

cout << "Queue is NOT Empty." << endl; break;

case 6:

if (q.isFull())

cout << "Queue is Full." << endl; else

cout << "Queue is NOT Full." << endl; break;

case 0:

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

default:

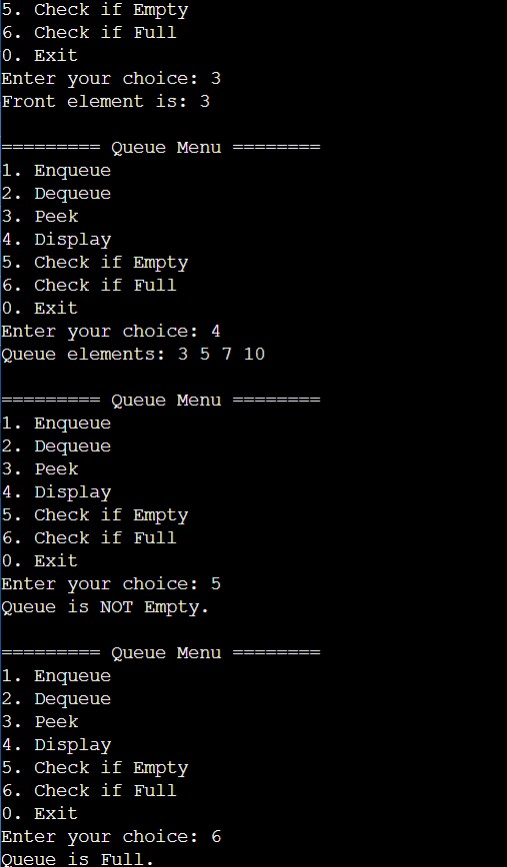
cout << "Invalid choice! Try again." << endl;

}

} while (choice != 0);

return 0;

}

Output: 

Ques2. Develop a menu driven program demonstrating the following operations on Circular Queues: enqueue(), dequeue(), isEmpty(), isFull(), display(), and peek().

Soln: #include <iostream> using namespace std; #define SIZE 5

class CircularQueue { private:

int arr[SIZE];

int front, rear;

public:

CircularQueue() { front = -1;

rear = -1;

}

bool isEmpty() {

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

}

bool isFull() {

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

}

void enqueue(int value) { if (isFull()) {

cout << "Queue Overflow! Cannot insert " << value << endl; return;

}

if (isEmpty()) {

front = rear = 0;

} else {

rear = (rear + 1) % SIZE;

}

arr[rear] = value;

cout << value << " enqueued into the queue." << endl;

}

void dequeue() { if (isEmpty()) {

cout << "Queue Underflow! Cannot dequeue." << endl; return;

}

cout << arr[front] << " dequeued from the queue." << endl; if (front == rear) {

front = rear = -1;

} else {

front = (front + 1) % SIZE;

}

}

void peek() {

if (isEmpty()) {

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

} else {

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

}

}

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) % SIZE;

}

cout << endl;

}

};

int main() { CircularQueue q; int choice, value;

do {

cout << "\n========= Circular Queue Menu ========" << endl; cout << "1. Enqueue" << endl;

cout << "2. Dequeue" << endl; cout << "3. Peek" << endl;

cout << "4. Display" << endl;

cout << "5. Check if Empty" << endl; cout << "6. Check if Full" << endl;

cout << "0. Exit" << endl;

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:

q.display(); break;

case 5:

if (q.isEmpty())

cout << "Queue is Empty." << endl; else

cout << "Queue is NOT Empty." << endl; break;

case 6:

if (q.isFull())

cout << "Queue is Full." << endl; else

cout << "Queue is NOT Full." << endl; break;

case 0:

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

default:

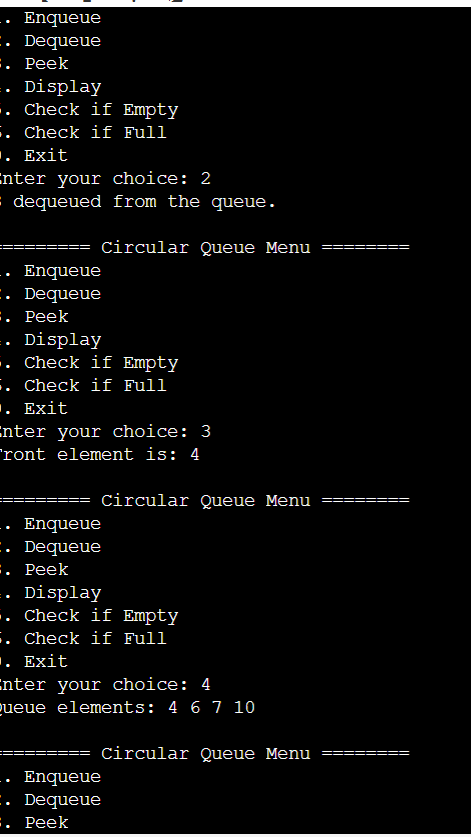
cout << "Invalid choice! Try again." << endl;

}

} while (choice != 0);

return 0;

}

Output: 

Ques3. 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

Soln : #include <iostream> using namespace std;

#define MAX 100

class Queue {

int arr[MAX];

int front, rear, size;

public:

Queue() {

front = 0;

rear = -1;

size = 0;

}

void enqueue(int x) { if (size == MAX) {

cout << "Queue Overflow\n"; return;

}

rear = (rear + 1) % MAX; arr[rear] = x;

size++;

}

int dequeue() { if (size == 0) {

cout << "Queue Underflow\n"; return -1;

}

int val = arr[front];

front = (front + 1) % MAX;

size--;

return val;

}

bool isEmpty() {

return size == 0;

}

int getSize() { return size;

}

int peek() {

if (size == 0) return -1; return arr[front];

}

};

void interleaveQueue(Queue &q) { int n = q.getSize();

if (n % 2 != 0) {

cout << "Queue size must be even!" << endl; return;

}

int half = n / 2; Queue firstHalf;

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

firstHalf.enqueue(q.dequeue());

}

// Interleave

while (!firstHalf.isEmpty()) {

q.enqueue(firstHalf.dequeue()); q.enqueue(q.dequeue());

}

}

int main() { Queue q;

q.enqueue(4);

q.enqueue(7); q.enqueue(11); q.enqueue(20); q.enqueue(5);

q.enqueue(9);

interleaveQueue(q);

cout << "Output: "; while (!q.isEmpty()) {

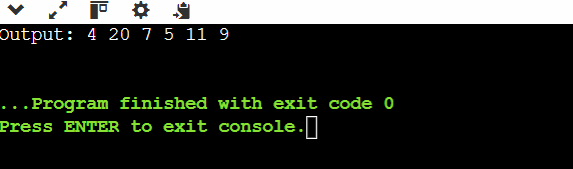
cout << q.dequeue() << " ";

}

cout << endl;

return 0;

}

Output:

Ques4. 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

Soln : #include <iostream> using namespace std;

class Queue { char arr[100]; int front, rear;

public:

Queue() {

front = 0;

rear = -1;

}

void enqueue(char x) { if (rear < 99) {

arr[++rear] = x;

}

}

void dequeue() { if (!empty()) {

front++;

}

}

char getFront() {

return arr[front];

}

bool empty() {

return front > rear;

}

};

void firstNonRepeating(string str) { int freq[256] = {0};

Queue q;

for (int i = 0; i < str.length(); i++) { char ch = str[i];

freq[ch]++;

q.enqueue(ch);

while (!q.empty() && freq[q.getFront()] > 1) { q.dequeue();

}

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

} else {

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

}

}

}

int main() {

string input;

getline(cin, input);

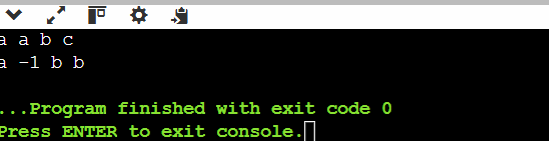
string filtered = "";

for (int i = 0; i < input.length(); i++) { if (input[i] != ' ') filtered += input[i];

}

firstNonRepeating(filtered); return 0;

}

Output:

Ques5. Write a program to implement a stack using (a) Two queues and (b) One Queue. Soln: #include <iostream>

using namespace std; #define SIZE 100 class Queue {

int arr[SIZE];

int front, rear; public:

Queue() {

front = rear = -1;

}

bool isEmpty() {

return (front == -1);

}

bool isFull() {

return (rear == SIZE - 1);

}

void enqueue(int x) { if (isFull()) {

cout << "Queue Overflow\n";

return;

}

if (front == -1) front = 0; arr[++rear] = x;

}

int dequeue() { if (isEmpty()) {

cout << "Queue Underflow\n"; return -1;

}

int val = arr[front]; if (front == rear)

front = rear = -1; else

front++; return val;

}

int peek() {

if (isEmpty()) return -1; return arr[front];

}

int size() {

if (isEmpty()) return 0; return rear - front + 1;

}

};

// Stack using Two Queues class StackTwoQ {

Queue q1, q2; public:

void push(int x) { q1.enqueue(x);

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

}

void pop() {

if (q1.isEmpty()) {

cout << "Stack Underflow\n"; return;

}

while (q1.size() > 1) {

q2.enqueue(q1.dequeue());

}

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

// swap q1 and q2 Queue temp = q1; q1 = q2;

q2 = temp;

}

void top() {

if (q1.isEmpty()) {

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

}

while (q1.size() > 1) {

q2.enqueue(q1.dequeue());

}

int val = q1.dequeue();

cout << "Top element: " << val << endl; q2.enqueue(val);

Queue temp = q1; q1 = q2;

q2 = temp;

}

bool empty() {

return q1.isEmpty();

}

};

int main() { StackTwoQ s; s.push(10);

s.push(20);

s.push(30);

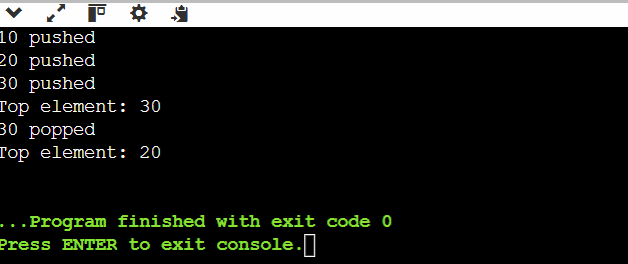
s.top();

s.pop();

s.top();

return 0;

}

Output: