# ASSIGNMENT-4 DATA STRUCTURES.

Name-Meet

Roll\_no-1024030343

Batch-2C24



Ques-1)

Code->

#include <iostream>

using namespace std;

class Queue {

private:

int front, rear, size;

int\* arr;

public:

Queue(int maxSize) {

size = maxSize;

arr = new int[size];

front = rear = -1;

}

~Queue() {

delete[] arr;

}

bool isEmpty() {

return front == -1;

}

bool isFull() {

return rear == size - 1;

}

void enqueue(int value) {

if (isFull()) {

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

return;

}

if (isEmpty()) {

front = rear = 0;

} else {

rear++;

}

arr[rear] = value;

cout << "Enqueued: " << value << endl;

}

void dequeue() {

if (isEmpty()) {

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

return;

}

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

if (front == rear) {

front = rear = -1; // Queue becomes empty

} else {

front++;

}

}

void peek() {

if (isEmpty()) {

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

} else {

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

}

}

void display() {

if (isEmpty()) {

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

return;

}

cout << "Queue contents: ";

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

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

}

cout << endl;

}

};

int main() {

int size;

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

cin >> size;

Queue q(size);

int choice, value;

do {

cout << "\nQueue Operations Menu\n";

cout << "1. Enqueue (Add)\n";

cout << "2. Dequeue (Remove)\n";

cout << "3. Check if Empty\n";

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

cout << "5. Display Queue\n";

cout << "6. Peek Front Element\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:

cout << (q.isEmpty() ? "Queue is empty.\n" : "Queue is not empty.\n");

break;

case 4:

cout << (q.isFull() ? "Queue is full.\n" : "Queue is not full.\n");

break;

case 5:

q.display();

break;

case 6:

q.peek();

break;

case 7:

cout << "Exiting program.\n";

break;

default:

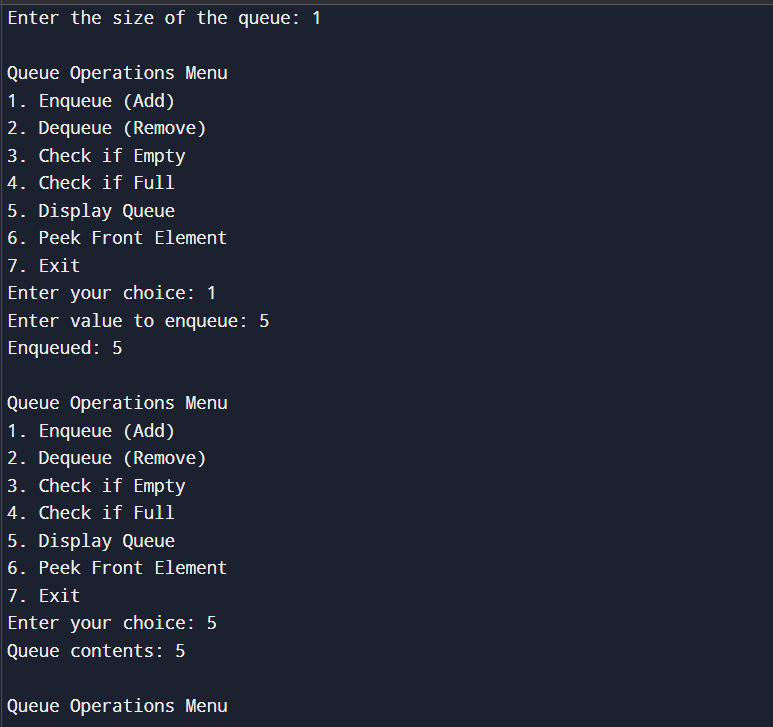
cout << "Invalid choice. Please try again.\n";

}

} while (choice != 7);

return 0;

}



Ques-2)

Code->

#include <iostream>

using namespace std;

class CircularQueue {

private:

int front, rear, size;

int\* arr;

public:

CircularQueue(int maxSize) {

size = maxSize;

arr = new int[size];

front = rear = -1;

}

~CircularQueue() {

delete[] arr;

}

bool isEmpty() {

return front == -1;

}

bool isFull() {

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

}

void enqueue(int value) {

if (isFull()) {

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

return;

}

if (isEmpty()) {

front = rear = 0;

} else {

rear = (rear + 1) % size;

}

arr[rear] = value;

cout << "Enqueued: " << value << endl;

}

void dequeue() {

if (isEmpty()) {

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

return;

}

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

if (front == rear) {

front = rear = -1; // Queue becomes empty

} else {

front = (front + 1) % size;

}

}

void peek() {

if (isEmpty()) {

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

} else {

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

}

}

void display() {

if (isEmpty()) {

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

return;

}

cout << "Queue contents: ";

int i = front;

while (true) {

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

if (i == rear) break;

i = (i + 1) % size;

}

cout << endl;

}

};

int main() {

int size;

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

cin >> size;

CircularQueue q(size);

int choice, value;

do {

cout << "\nCircular Queue Operations Menu\n";

cout << "1. Enqueue (Add)\n";

cout << "2. Dequeue (Remove)\n";

cout << "3. Check if Empty\n";

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

cout << "5. Display Queue\n";

cout << "6. Peek Front Element\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:

cout << (q.isEmpty() ? "Queue is empty.\n" : "Queue is not empty.\n");

break;

case 4:

cout << (q.isFull() ? "Queue is full.\n" : "Queue is not full.\n");

break;

case 5:

q.display();

break;

case 6:

q.peek();

break;

case 7:

cout << "Exiting program.\n";

break;

default:

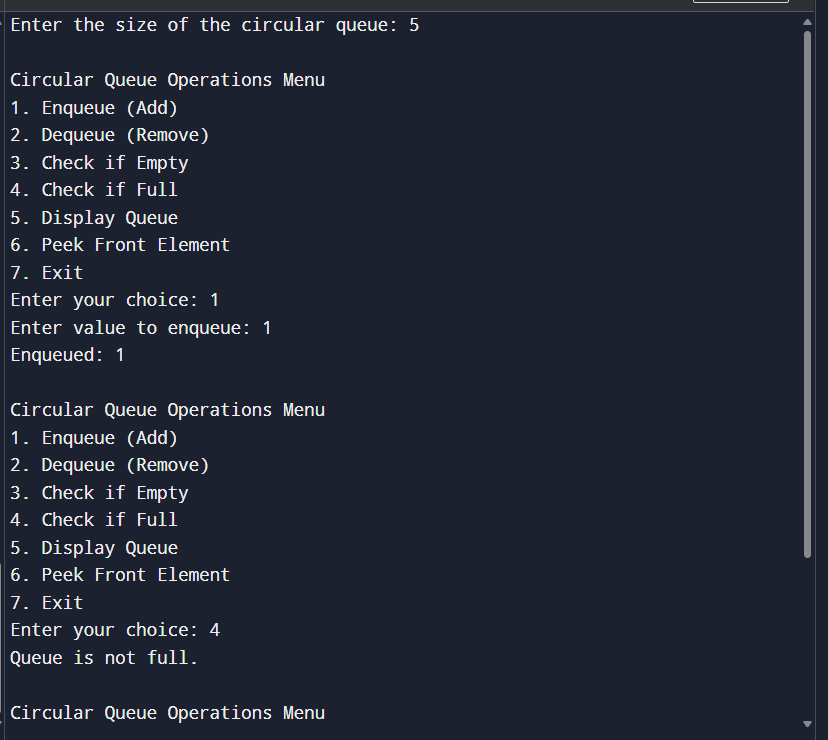
cout << "Invalid choice. Please try again.\n";

}

} while (choice != 7);

return 0;

}



Ques-3)

Code->

#include <iostream>

#include <queue>

using namespace std;

void interleaveQueue(queue<int>& q) {

int n = q.size();

if (n % 2 != 0) {

cout << "Queue must have even number of elements.\n";

return;

}

int half = n / 2;

queue<int> firstHalf;

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

firstHalf.push(q.front());

q.pop();

}

while (!firstHalf.empty()) {

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

firstHalf.pop();

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

q.pop();

}

cout << endl;

}

int main() {

queue<int> q;

q.push(4);

q.push(7);

q.push(11);

q.push(20);

q.push(5);

q.push(9);

cout << "Original queue: ";

queue<int> temp = q;

while (!temp.empty()) {

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

temp.pop();

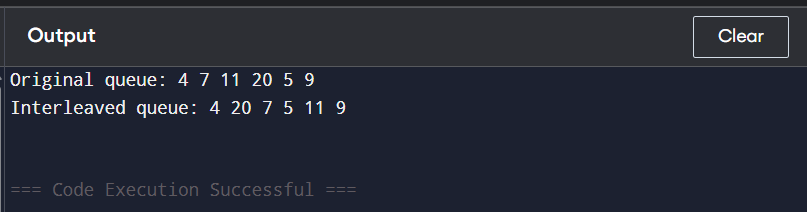
}

cout << "\nInterleaved queue: ";

interleaveQueue(q);

return 0;

}



Ques-4)

Code->

#include <iostream>

#include <queue>

using namespace std;

void firstNonRepeating(string str) {

queue<char> q;

int freq[26] = {0};

for (int i = 0; i < str.length(); i++) {

char ch = str[i];

freq[ch - 'a']++;

q.push(ch);

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

q.pop();

}

if (q.empty()) {

cout << "-1 ";

} else {

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

}

}

cout << endl;

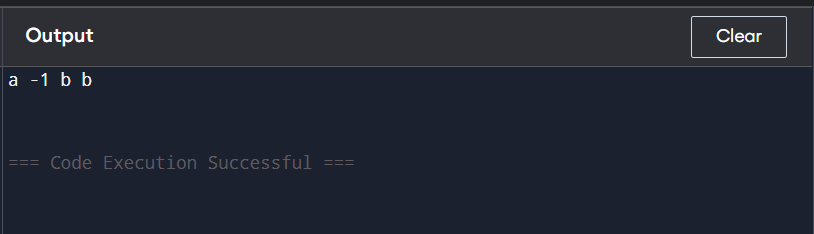
}

int main() {

string input = "aabc";

firstNonRepeating(input);

return 0;

}

Ques-5)

1)using two queues

#include <iostream>

#include <queue>

using namespace std;

class StackTwoQueues {

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()) q1.pop();

}

int top() {

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

return q1.front();

}

bool isEmpty() {

return q1.empty();

}

void display() {

queue<int> temp = q1;

while (!temp.empty()) {

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

temp.pop();

}

cout << endl;

}

};

int main() {

StackTwoQueues st;

st.push(10);

st.push(20);

st.push(30);

st.display();

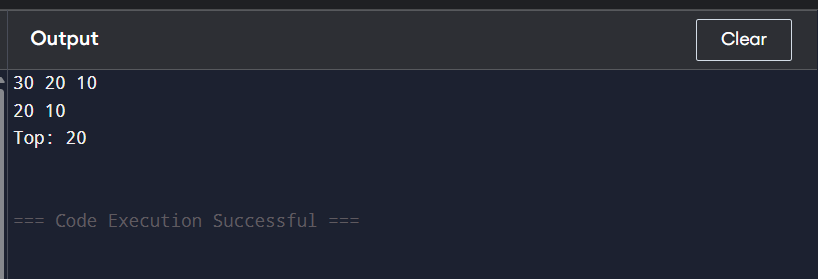
st.pop();

st.display();

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

return 0;

}



Using one queue

#include <iostream>

#include <queue>

using namespace std;

class StackOneQueue {

private:

queue<int> q;

public:

void push(int x) {

q.push(x);

int n = q.size();

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

q.push(q.front());

q.pop();

}

}

void pop() {

if (q.empty()) {

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

return;

}

q.pop();

}

int top() {

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

return q.front();

}

bool isEmpty() {

return q.empty();

}

void display() {

queue<int> temp = q;

while (!temp.empty()) {

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

temp.pop();

}

cout << endl;

}

};

int main() {

StackOneQueue st;

st.push(100);

st.push(200);

st.push(300);

st.display();

st.pop();

st.display();

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

return 0;

}

