Name memansi  
subgroup 2c35  
roll no 1024030006

Assignment 4  
1) Develop a menu driven program demonstrating the following operations on simple Queues: enqueue(), dequeue(), isEmpty(), isFull(), display(), and peek().

Sol #include <iostream>

using namespace std;

class SimpleQueue {

int \*arr;

int front, rear, size, capacity;

public:

SimpleQueue(int c) {

capacity = c;

arr = new int[c];

front = 0;

rear = -1;

size = 0;

}

bool isEmpty() {

return size == 0;

}

bool isFull() {

return size == capacity;

}

void enqueue(int x) {

if (isFull()) {

cout << "Queue is Full\n";

return;

}

rear = (rear + 1) % capacity;

arr[rear] = x;

size++;

}

int dequeue() {

if (isEmpty()) {

cout << "Queue is Empty\n";

return -1;

}

int x = arr[front];

front = (front + 1) % capacity;

size--;

return x;

}

int peek() {

if (isEmpty()) {

cout << "Queue is Empty\n";

return -1;

}

return arr[front];

}

void display() {

if (isEmpty()) {

cout << "Queue is Empty\n";

return;

}

int i = front;

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

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

i = (i + 1) % capacity;

}

cout << endl;

}

};

int main() {

int c;

cout << "Enter size of queue: ";

cin >> c;

SimpleQueue q(c);

while (true) {

cout << "\n1.Enqueue 2.Dequeue 3.Peek 4.Display 5.Exit\n";

int ch;

cin >> ch;

if (ch == 1) {

int x;

cout << "Enter item: ";

cin >> x;

q.enqueue(x);

} else if (ch == 2) {

cout << "Dequeued: " << q.dequeue() << endl;

} else if (ch == 3) {

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

} else if (ch == 4) {

q.display();

} else if (ch == 5) {

break;

} else {

cout << "Invalid choice\n";

}

}

return 0;

}  
  


2) Develop a menu driven program demonstrating the following operations on Circular Queues: enqueue(), dequeue(), isEmpty(), isFull(), display(), and peek().  
sol #include <iostream>

using namespace std;

class CircularQueue {

int \*arr;

int front, rear, size, capacity;

public:

CircularQueue(int c) {

capacity = c;

arr = new int[c];

front = -1;

rear = -1;

size = 0;

}

bool isEmpty() {

return size == 0;

}

bool isFull() {

return size == capacity;

}

void enqueue(int x) {

if (isFull()) {

cout << "Queue is Full\n";

return;

}

if (front == -1) front = 0;

rear = (rear + 1) % capacity;

arr[rear] = x;

size++;

}

int dequeue() {

if (isEmpty()) {

cout << "Queue is Empty\n";

return -1;

}

int x = arr[front];

if (front == rear) {

front = rear = -1;

size = 0;

} else {

front = (front + 1) % capacity;

size--;

}

return x;

}

int peek() {

if (isEmpty()) {

cout << "Queue is Empty\n";

return -1;

}

return arr[front];

}

void display() {

if (isEmpty()) {

cout << "Queue is Empty\n";

return;

}

int i = front;

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

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

i = (i + 1) % capacity;

}

cout << endl;

}

};

int main() {

int c;

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

cin >> c;

CircularQueue q(c);

while (true) {

cout << "\n1.Enqueue 2.Dequeue 3.Peek 4.Display 5.Exit\n";

int ch;

cin >> ch;

if (ch == 1) {

int x;

cout << "Enter item: ";

cin >> x;

q.enqueue(x);

} else if (ch == 2) {

cout << "Dequeued: " << q.dequeue() << endl;

} else if (ch == 3) {

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

} else if (ch == 4) {

q.display();

} else if (ch == 5) {

break;

} else {

cout << "Invalid choice\n";

}

}

return 0;

}



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

Sol #include <iostream>

#include <queue>

using namespace std;

void interleaveQueue(queue<int>& q) {

int n = q.size();

if (n % 2 != 0) 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 arr[] = {4, 7, 11, 20, 5, 9};

int n = 6;

for (int i = 0; i < n; i++) q.push(arr[i]);

interleaveQueue(q);

while (!q.empty()) {

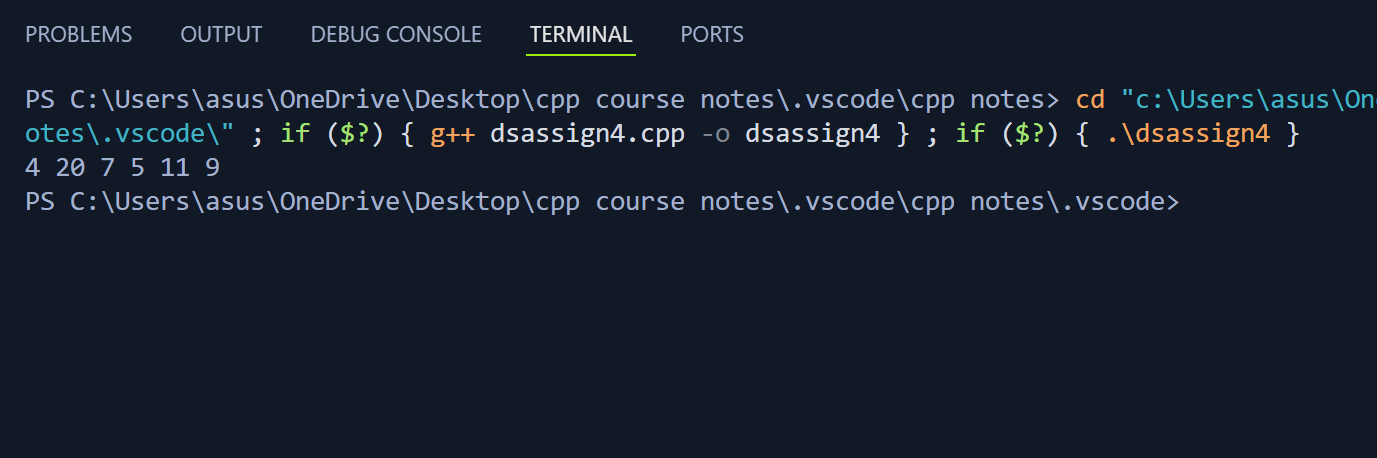
cout << q.front() << " ";

q.pop();

}

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;

void firstNonRepeating(string s) {

unordered\_map<char, int> freq;

queue<char> q;

for (char ch : s) {

freq[ch]++;

q.push(ch);

while (!q.empty() && freq[q.front()] > 1) q.pop();

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

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

}

}

int main() {

string s;

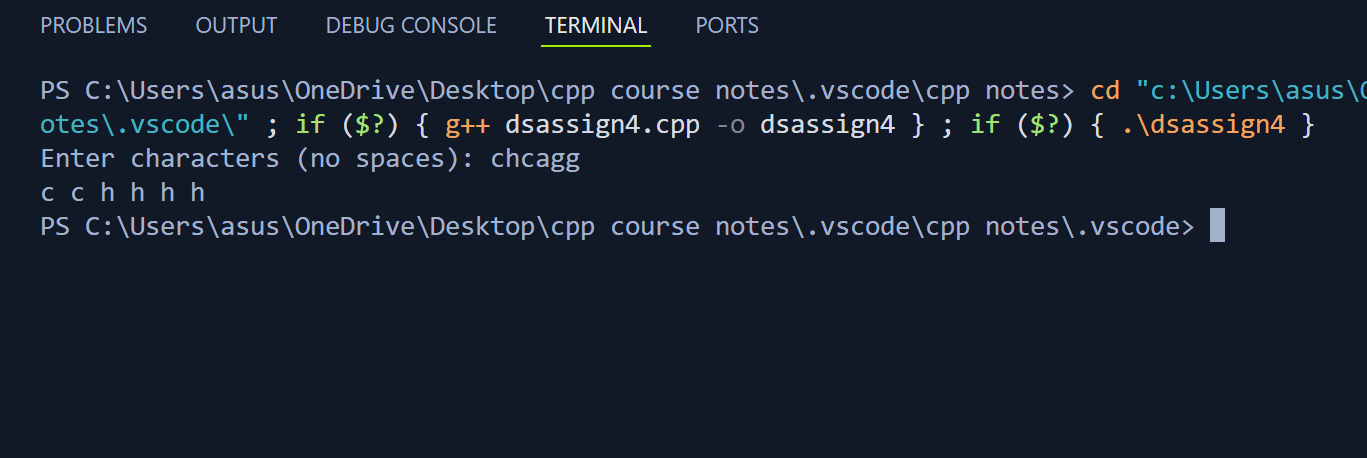
cout << "Enter characters (no spaces): ";

cin >> s;

firstNonRepeating(s);

return 0;

}



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

Sol #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);

}

int pop() {

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

int x = q1.front();

q1.pop();

return x;

}

int top() {

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

return q1.front();

}

bool empty() {

return q1.empty();

}

};

int main() {

StackTwoQueues st;

st.push(10);

st.push(20);

st.push(30);

cout << st.pop() << endl;

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

return 0;

}

