# DSA LAB 05

NAME: Kashan BAIG

Section: D

ROLL NO: CT-178

**QUESTION 1**

**INPUT:**

#include <iostream>

using namespace std;

class circulardeq{

private:

int front,rear, size;

int\* arr;

public:

circulardeq(int k): size(k), front(-1), rear(-1){

arr = new int[size];

}

bool full(){

if((front == 0 && rear == size-1) || (front == rear + 1))

return true;

return false;

}

bool empty(){

if(front == -1) return true;

return false;

}

bool insertfront(int value){

if(full()) {

cout<<"The queue is FULL!"<<endl;

return false;

}

if(front == -1)

front = rear = 0;

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

else front--;

arr[front] = value;

return true;

}

bool insertlast(int value){

if(full()) {

cout<<"The queue is FULL!"<<endl;

return false;}

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

else if(rear == size-1)

rear = 0;

else rear++;

arr[rear] = value;

return true;

}

bool delfront(){

if(empty()) return false;

cout<<"Deleted "<<arr[front]<<endl;

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

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

else front++;

return true;

}

bool dellast(){

if(empty()) return false;

cout<<"Deleted "<<arr[rear]<<endl;

if(front == rear)

front = rear = -1;

else if(rear == 0) rear = size-1;

else rear--;

return true;

}

int getfront(){

if(empty()) return -1;

return arr[front]; }

int getrear(){

if(empty()) return -1;

return arr[rear];

}

void Display(){

int i = front;

while(true){

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

if(i == rear) break;

i = (i+1)%size;}

cout<<endl; }

};

int main(){

circulardeq circulardeq(3);

cout<<circulardeq.insertlast(1)<<endl;

cout<<circulardeq.insertlast(2)<<endl;

cout<<circulardeq.insertfront(3)<<endl;

cout<<circulardeq.insertfront(4)<<endl;

cout<<circulardeq.getrear()<<endl;

cout<<circulardeq.full()<<endl;

cout<<circulardeq.dellast()<<endl;

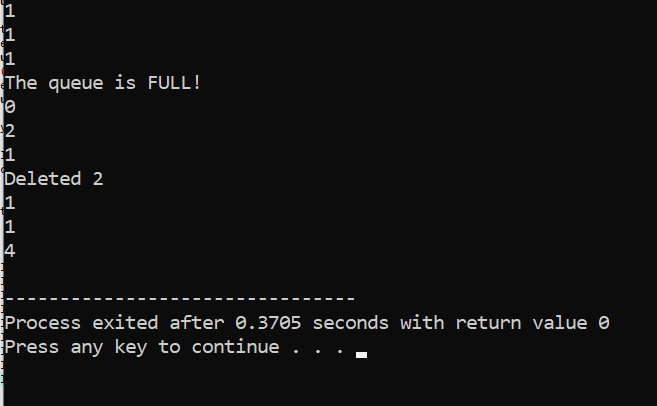
cout<<circulardeq.insertfront(4)<<endl;

cout<<circulardeq.getfront()<<endl;

return 0;

}

**Output:**

****

**QUESTION 2**

**INPUT:**

#include <iostream>

#include <queue>

using namespace std;

int winner(int n, int k){

queue<int> q;

for (int i=1;i<= n;i++)

q.push(i);

while (q.size() > 1) {

for (int i = 1; i < k; i++) {

q.push(q.front());

q.pop();}

q.pop();

}

return q.front();}

int main(){

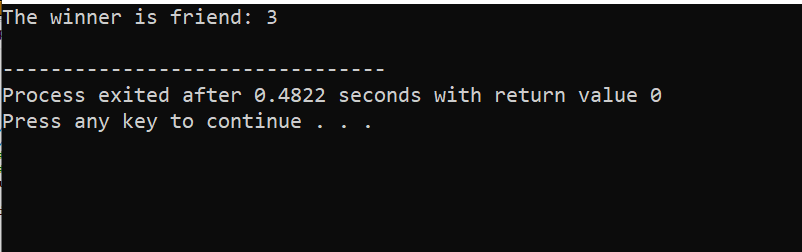
int n= 5, k= 2;

cout<<"The winner is friend: "<<winner(n, k)<<endl;

return 0;

}

**Output:**

****

**QUESTION 3**

**INPUT:**

#include <iostream>

#include <queue>

#include <vector>

#include <string>

using namespace std;

class Call {

public:

int callid;

int arrivaltime;

string name;

Call(int id, int time, string name) {

callid = id;

arrivaltime = time;

this->name = name;

}

};

class callcenter{

queue<Call> callque;

int numcrs;

vector<bool> available;

public:

callcenter(int csrs){

numcrs = csrs;

available = vector<bool>(csrs, true);}

void add(Call c){

callque.push(c);

cout << "Call from " << c.name << " (ID "<<c.callid<<")added to queue at time "<<c.arrivaltime<<endl;

}

void process() {

int currtime = 0;

while (!callque.empty()) {

for (int i=0;i<numcrs && !callque.empty();i++) {

if (available[i]) {

Call c = callque.front();

callque.pop();

available[i] = false;

cout << "CSR " <<i+1<<" started handling call ID "<< c.callid << " from "<<c.name << " at time " << currtime << endl;

currtime+=2;

available[i] = true;

cout<<"CSR " <<i+1<<" finished call ID "<<c.callid<<" at time "<<currtime<<endl;}

}

currtime++;}

}

};

int main(){

callcenter c(2);

c.add(Call(1, 0,"Ali"));

c.add(Call(2, 1,"laiba"));

c.add(Call(3, 2,"Sara"));

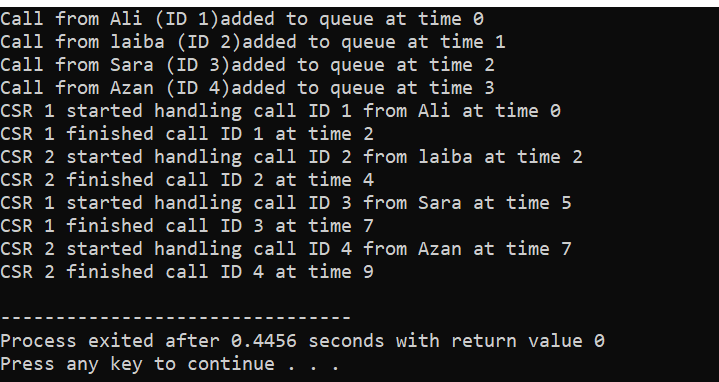
c.add(Call(4, 3,"Azan"));

c.process();

return 0;

}

**Output:**

****

**QUESTION 4**

**INPUT:**

#include <iostream>

#include <queue>

using namespace std;

class productofnum {

queue<int> stream;

public:

productofnum() {}

void add(int num) {

stream.push(num);}

int getproduct(int k) {

if (k>stream.size()) {

cout<<"Not enough elements in the stream"<<endl;

return 0;}

int product = 1;

queue<int> temp = stream;

int start = temp.size()-k;

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

temp.pop();

}

while (!temp.empty()) {

product \*= temp.front();

temp.pop();}

return product;}

};

int main() {

productofnum stream;

stream.add(3);

stream.add(0);

stream.add(2);

stream.add(5);

stream.add(4);

cout << stream.getproduct(2) << endl;

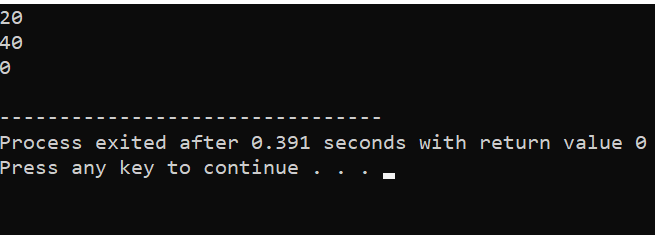
cout << stream.getproduct(3) << endl;

cout << stream.getproduct(4) << endl;

return 0;

}

**Output:**



**QUESTION 5**

**INPUT:**

#include <iostream>

#include <queue>

using namespace std;

class Datastream{

int value, k;

queue<int> q;

int count;

public:

Datastream(int v, int k){

value = v;

this->k=k;

count=0;

}

bool consec(int num){

q.push(num);

if (num == value) count++;

if (q.size() > k) {

if (q.front()==value) count--;

q.pop();

}

return (q.size() == k && count == k);}

};

int main() {

Datastream ds(4,3);

cout<<ds.consec(4)<<endl;

cout<<ds.consec(4)<<endl;

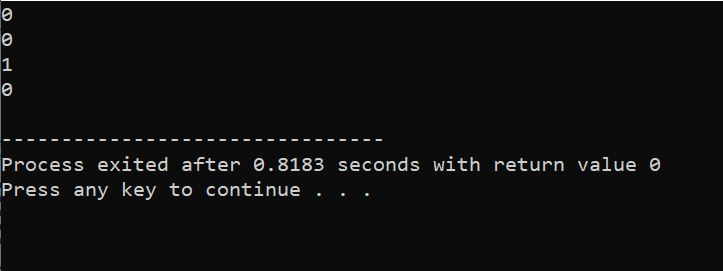
cout<<ds.consec(4)<<endl;

cout<<ds.consec(3)<<endl;

return 0;

}

**Output:**

****