NAME: MOAZZAM FAROOQUI
ROLLNO: CT-24068
COURSE CODE: CT-159
ASSIGNMENT: DSA LAB#05
INSTRUCTOR: SAYYDA SAHAR FATIMA

Q1.

SOURCE CODE:

```cpp
#include<bits/stdc++.h>
using namespace std;

class MyCircularDeque{
private:
    vector<int> dq;
    int front;
    int rear;
    int size;
    int capacity;

public:
    MyCircularDeque(int k){
        capacity=k;
        dq.resize(k);
        front=0;
        rear=0;
        size=0;
    }

    bool insertFront(int value){
        if(size==capacity) return false;
        front=(front-1+capacity)%capacity;
        dq[front]=value;
        size++;
        return true;
    }
```

```
    bool insertLast(int value){
        if(size==capacity) return false;
        dq[rear]=value;
        rear=(rear+1)%capacity;
        size++;
        return true;
    }

    bool deleteFront(){
        if(size==0) return false;
        front=(front+1)%capacity;
        size--;
        return true;
    }

    bool deleteLast(){
        if(size==0) return false;
        rear=(rear-1+capacity)%capacity;
        size--;
        return true;
    }

    int getFront(){
        if(size==0) return -1;
        return dq[front];
    }
```

```
    int getRear(){
        if(size==0) return -1;
        return dq[(rear-1+capacity)%capacity];
    }

    bool isEmpty(){
        return size==0;
    }

    bool isFull(){
        return size==capacity;
    }
};
```

```
int main(void){
    MyCircularDeque myCircularDeque(3);
    cout<<myCircularDeque.insertLast(1)<<endl;
    cout<<myCircularDeque.insertLast(2)<<endl;
    cout<<myCircularDeque.insertFront(3)<<endl;
    cout<<myCircularDeque.insertFront(4)<<endl;
    cout<<myCircularDeque.getRear()<<endl;
    cout<<myCircularDeque.isFull()<<endl;
    cout<<myCircularDeque.deleteLast()<<endl;
    cout<<myCircularDeque.insertFront(4)<<endl;
    cout<<myCircularDeque.getFront()<<endl;

    return 0;
}
```

OUTPUT:

```
1
1
1
0
2
1
1
1
4

--------------------------------
Process exited after 0.342 seconds with return value 0
Press any key to continue . . .
```

Q2.

SOURCE CODE:

```cpp
#include<bits/stdc++.h>
using namespace std;

int findthewinner(int n,int k){
    int winner=0;
    for(int i=2;i<=n;i++){
        winner=(winner+k)%i;
    }
    return winner+1;
}

int main(void){
    int n;
    int k;
    cout<<"ENTER NUMBER OF FRIENDS:"<<endl;
    cin>>n;
    cout<<"ENTER THE NUMBER THAT YOU COUNTED(K):"<<endl;
    cin>>k;
    int result;
    result=findthewinner(n,k);
    cout<<"THE WINNER IS FRIEND "<<result<<endl;
    return 0;
}
```

OUTPUT:

```
ENTER NUMBER OF FRIENDS:
4
ENTER THE NUMBER THAT YOU COUNTED(K):
3
THE WINNER IS FRIEND 1

--------------------------------
Process exited after 6.02 seconds with return value 0
Press any key to continue . . .
```

Q3.

SOURCE CODE:

```
#include<bits/stdc++.h>
using namespace std;

class Call{
private:
    int callid,hour,minute;
    string customername;
public:
    Call(int ci,int h,int m,string cn){
        callid=ci;
        hour=h;
        minute=m;
        customername=cn;
    }
    int getid(){
    return callid;
    }
    int gethour(){return hour;}
    int getminute(){return minute;}
    string getname(){return customername;}
    int gettimeinminutes(){return hour*60+minute;}
};
```

```
class Callcenter{
private:
    queue<Call> callqueue;
    int numcsrs,currenttime;
    vector<bool> csravailable;
public:
    Callcenter(int csrs){
        numcsrs=csrs;
        csravailable.resize(csrs,true);
        currenttime=0;
    }
    void addcall(int id,int h,int m,string name){
        Call newcall(id,h,m,name);
        callqueue.push(newcall);
        cout<<"CALL ADDED-ID:"<<id<<" NAME:"<<name<<" ARRIVAL:";
        cout<<setw(2)<<setfill('0')<<h<<":";
        cout<<setw(2)<<setfill('0')<<m<<endl;
    }
```

```
    void processcall(){
        if(!callqueue.empty()) currenttime=callqueue.front().gettimeinminutes();
        while(!callqueue.empty()){
            for(int i=0;i<numcsrs;i++) csravailable[i]=true;
            int assigned=0;
            for(int i=0;i<numcsrs;i++){
                if(!callqueue.empty()){
                    Call currentcall=callqueue.front();
                    if(csravailable[i] && currentcall.gettimeinminutes()<=currenttime){
                        callqueue.pop();
                        csravailable[i]=false;
                        assigned++;
                        int h=currenttime/60;
                        int m=currenttime%60;
                        cout<<"CSR "<<i+1<<" is handling Call ID "<<currentcall.getid()
                            <<" from "<<currentcall.getname()<<" at time ";
                        cout<<setw(2)<<setfill('0')<<h<<":";
                        cout<<setw(2)<<setfill('0')<<m<<endl;
                    }
                }
            }
            if(assigned==0){
                if(!callqueue.empty()){
                    int nextCallTime=callqueue.front().gettimeinminutes();
                    if(nextCallTime>currenttime) currenttime=nextCallTime;
                    else currenttime++;
                }
            }else currenttime++;
        }
    }
};

int main(){
    Callcenter center(3);
    center.addcall(1,8,5,"Moazzam");
    center.addcall(2,8,41,"Ali");
    center.addcall(3,9,41,"Danyal");
    center.addcall(4,9,42,"Husn");
    center.addcall(5,10,43,"Saad");
    center.processcall();
    return 0;
}
```

OUTPUT:

```
CALL ADDED-ID:1 NAME:Moazzam ARRIVAL:08:05
CALL ADDED-ID:2 NAME:Ali ARRIVAL:08:41
CALL ADDED-ID:3 NAME:Danyal ARRIVAL:09:41
CALL ADDED-ID:4 NAME:Husn ARRIVAL:09:42
CALL ADDED-ID:5 NAME:Saad ARRIVAL:10:43
CSR 1 is handling Call ID 1 from Moazzam at time 08:05
CSR 1 is handling Call ID 2 from Ali at time 08:41
CSR 1 is handling Call ID 3 from Danyal at time 09:41
CSR 1 is handling Call ID 4 from Husn at time 09:42
CSR 1 is handling Call ID 5 from Saad at time 10:43

--------------------------------
Process exited after 0.4157 seconds with return value 0
Press any key to continue . . .
```

Q4.

SOURCE CODE:

```cpp
#include<bits/stdc++.h>
using namespace std;

class ProductOfNumbers{
private:
    vector<int> nums;
public:
    ProductOfNumbers(){}
    void add(int num){
        nums.push_back(num);
    }
    int getProduct(int k){
        int product=1;
        int n=nums.size();
        for(int i=n-k;i<n;i++){
            product*=nums[i];
        }
        return product;
    }
};
```

```cpp
int main(void){
    ProductOfNumbers productOfNumbers;
    productOfNumbers.add(3);
    productOfNumbers.add(0);
    productOfNumbers.add(2);
    productOfNumbers.add(5);
    productOfNumbers.add(4);
    cout<<productOfNumbers.getProduct(2)<<endl;
    cout<<productOfNumbers.getProduct(3)<<endl;
    cout<<productOfNumbers.getProduct(4)<<endl;
    productOfNumbers.add(8);
    cout<<productOfNumbers.getProduct(2)<<endl;
    return 0;
}
```

OUTPUT:

```
20
40
0
32

--------------------------------
Process exited after 0.3187 seconds with return value 0
Press any key to continue . . .
```

Q5.

SOURCE CODE:

```
#include<bits/stdc++.h>
using namespace std;

class DataStream{
    private:
        vector<int>datastream;
        int value;
        int k;
    public:
        DataStream(int val,int kk){
            value=val;
            k=kk;
        }
        bool consec(int num){
            datastream.push_back(num);
        if(datastream.size()>k){
            datastream.erase(datastream.begin());
        }
        if(datastream.size()<k){
            return false;
        }
        for(int i=0;i<k;i++){
            if(datastream[i]!=value){
                return false;
            }
        }
            return true;
            }
};

int main(void){
    DataStream m(4,3);
    cout<<m.consec(4)<<endl;
    cout<<m.consec(4)<<endl;
    cout<<m.consec(4)<<endl;
    cout<<m.consec(3)<<endl;
    return 0;
}
```

OUTPUT:

```
0
0
1
0

--------------------------------
Process exited after 0.6435 seconds with return value 0
Press any key to continue . . .
```