LAB

REPORT

CSE 114 : Data Structure and Algorithms Sessional

|  |  |
| --- | --- |
| PREPARED BY  Mehrin Farzana  ID: 2101013  Session: 2021-2022  Date: 4/10/2023 | SUPERVISED BY  Suman Saha  Lecturer  Department of IRE, BDU |

|  |  |
| --- | --- |
| logo1 | BANGABANDHU SHEIKH MUJIBUR RAHMAN DIGITAL UNIVERSITY  (BDU) |
|  |  |

**List of Problems**

1. Implement circular queue.
2. Implement a program to solve the Tower of Hanoi problem for n disks, and print the sequence of moves required to transfer all the disks from one peg to another while adhering to the rules.
3. Write down a program to Check Whether a Number is Palindrome or Not.

**Problem No.:** 01

**Problem Statement:**

Implement circular queue.

**Code:**

#include <stdio.h>

int rear = -1 , front = -1;

int is\_full(int \*a, int n){

if(front==0 && rear==n-1){

return 1;

}

if(rear==front-1){

return 1;

}

return 0;

}

int is\_empty(int \*a, int n){

if(front==-1){

return 1;

}

return 0;

}

void insert(int \*a, int n, int item){

if(is\_full(a,n)){

printf("Overflow!\n");

return;

}

else if(front==-1){

front++;

rear++;

}

else if(rear<n-1){

rear++;

}

else if(rear==n-1){

rear=0;

}

a[rear]=item;

}

void delete(int \*a, int n){

if(front==-1){

printf("Underflow!\n");

return;

}

if(front==rear){

front=-1;

rear=-1;

// printf("%d %d\n", front, rear);

}

else if(front==n-1){

front=0;

}

else{

front++;

}

}

void display(int \*a, int n){

printf("front: %d , rear: %d\n", front, rear);

if(front==-1 && rear==-1){

printf("Empty queue\n");

}

else if(front<=rear){

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

printf("%d ", a[i]);

}

}

else if(front>rear){

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

printf("%d ", a[i]);

}

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

printf("%d ", a[i]);

}

}

printf("\n");

}

int main() {

int b=1;

int n,item;

printf("Enter queue size: ");

scanf("%d", &n);

int a[n];

while(b){

printf("0.Exit\n1. Insert\n2. Delete\n3. Display\n");

scanf("%d", &b);

switch(b){

case 0:

break;

case 1:

printf("Item: ");

scanf("%d", &item);

insert(a, n, item);

break;

case 2:

delete(a, n);

break;

case 3:

display(a, n);

break;

default:

break;

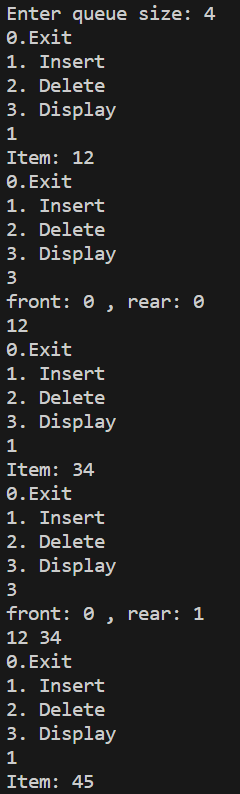
}

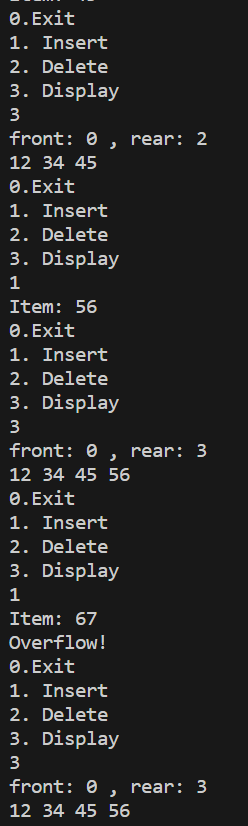
}

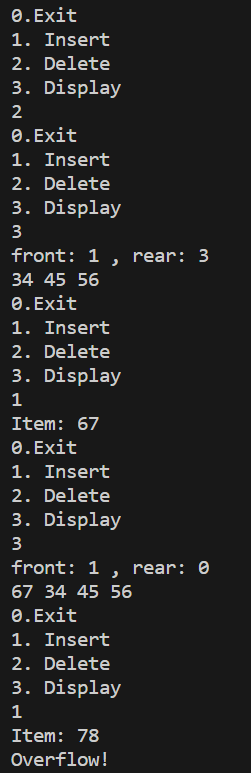
return 0;

}

**Output:**







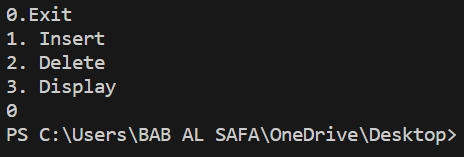


Fig 1.1: Output on console for case 1.

**Problem No.:** 02

**Problem Statement:**

Implement a program to solve the Tower of Hanoi problem for n disks, and print the sequence of moves required to transfer all the disks from one peg to another while adhering to the rules.

**Code:**

#include <stdio.h>

void towerOfHanoi(int n, char from\_rod, char to\_rod, char aux\_rod)

{

if (n == 1)

{

printf("\n Move disk 1 from rod %c to rod %c", from\_rod, to\_rod);

return;

}

towerOfHanoi(n-1, from\_rod, aux\_rod, to\_rod);

printf("\n Move disk %d from rod %c to rod %c", n, from\_rod, to\_rod);

towerOfHanoi(n-1, aux\_rod, to\_rod, from\_rod);

}

int main()

{

int n;

scanf("%d", &n);

towerOfHanoi(n, 'A', 'C', 'B');

return 0;

}

**Output:**

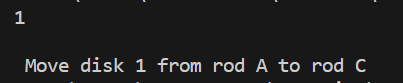


Fig 1.1: Output on console for case 1.

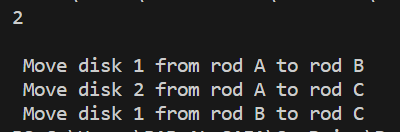


Fig 1.2: Output on console for case 2.

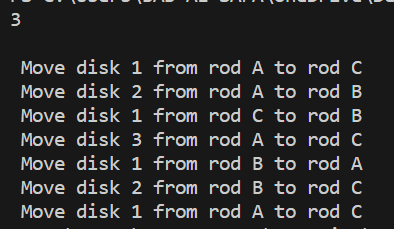


Fig 1.3: Output on console for case 3.

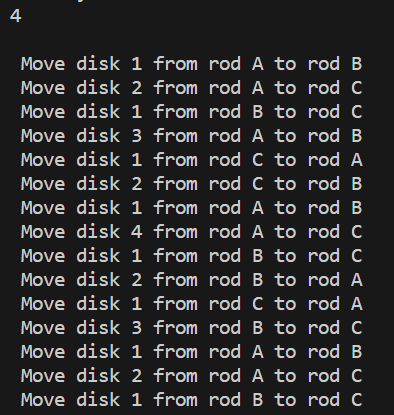


Fig 1.4: Output on console for case 4.

**Problem No.:** 03

**Problem Statement:**

Write down a program to Check Whether a Number is Palindrome or Not.

**Code:**

#include <stdio.h>

#include <string.h>

int main() {

char s[101];

scanf("%s", s);

int n = strlen(s), flag=0;

for(int i=0; i<=n/2; i++){

if(s[i]!=s[n-1-i])

flag=1;

}

if(!flag)

printf("Pallindrome!");

else if(flag)

printf("Not a Pallindrome!");

return 0;

}

**Output:**



Fig 1.1: Output on console for case 1.



Fig 1.2: Output on console for case 2.



Fig 1.3: Output on console for case 3.



Fig 1.4: Output on console for case 4.