Mohd. Aftab Alam
02013302717

EXPERIMENT 3

Aim :

Write a program to implement iterative Tower of Hanoi.

Code :

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <limits.h>

struct Stack
{
    unsigned capacity;
    int top;
    int *array;
};

struct Stack *createStack(unsigned capacity)
{
    struct Stack *stack = (struct Stack *)malloc(sizeof(struct Stack));
    stack->capacity = capacity;
    stack->top = -1;
    stack->array = (int *)malloc(stack->capacity * sizeof(int));
    return stack;
}

int isFull(struct Stack *stack)
{
    return (stack->top == stack->capacity - 1);
}

int isEmpty(struct Stack *stack)
{
    return (stack->top == -1);
}

void push(struct Stack *stack, int item)
{
    if (isFull(stack))
        return;
    stack->array[++stack->top] = item;
}
```

```c
int pop(struct Stack *stack)
{
    if (isEmpty(stack))
        return INT_MIN;
    return stack->array[stack->top--];
}

void moveDisk(char fromPeg, char toPeg, int disk)
{
printf(" Move the Disk %d from \'%c\' to \'%c\'\n", disk, fromPeg, toPeg);
}

void moveDisksBetweenTwoPoles(struct Stack *src, struct Stack *dest, char
s, char d)
{
    int pole1TopDisk = pop(src);
    int pole2TopDisk = pop(dest);

    if (pole1TopDisk == INT_MIN)
    {
        push(src, pole2TopDisk);
        moveDisk(d, s, pole2TopDisk);
    }

    else if (pole2TopDisk == INT_MIN)
    {
        push(dest, pole1TopDisk);
        moveDisk(s, d, pole1TopDisk);
    }

    else if (pole1TopDisk > pole2TopDisk)
    {
        push(src, pole1TopDisk);
        push(src, pole2TopDisk);
        moveDisk(d, s, pole2TopDisk);
    }

    else
    {
        push(dest, pole2TopDisk);
        push(dest, pole1TopDisk);
        moveDisk(s, d, pole1TopDisk);
    }
}
```

```
void tohIterative(int num_of_disks, struct Stack *src, struct Stack *aux,
struct Stack *dest)
{
    int i, total_num_of_moves;
    char s = 'S', d = 'D', a = 'A';

    if (num_of_disks % 2 == 0)
    {
        char temp = d;
        d = a;
        a = temp;
    }
    total_num_of_moves = pow(2, num_of_disks) - 1;

    for (i = num_of_disks; i >= 1; i--)
        push(src, i);

    for (i = 1; i <= total_num_of_moves; i++)
    {
        if (i % 3 == 1)
            moveDisksBetweenTwoPoles(src, dest, s, d);

        else if (i % 3 == 2)
            moveDisksBetweenTwoPoles(src, aux, s, a);

        else if (i % 3 == 0)
            moveDisksBetweenTwoPoles(aux, dest, a, d);
    }
}

int main()
{
    unsigned num_of_disks;
    printf("\n Enter Number of Disks : ");
    scanf("%d", &num_of_disks);

    struct Stack *src, *dest, *aux;

    src = createStack(num_of_disks);
    aux = createStack(num_of_disks);
    dest = createStack(num_of_disks);

    tohIterative(num_of_disks, src, aux, dest);
    return 0;
}
```

Output Screenshot :