DAA LAB TASK 3

1. Converting recursive programs to non-recursive programs. Towers of Hanoi Problem example.

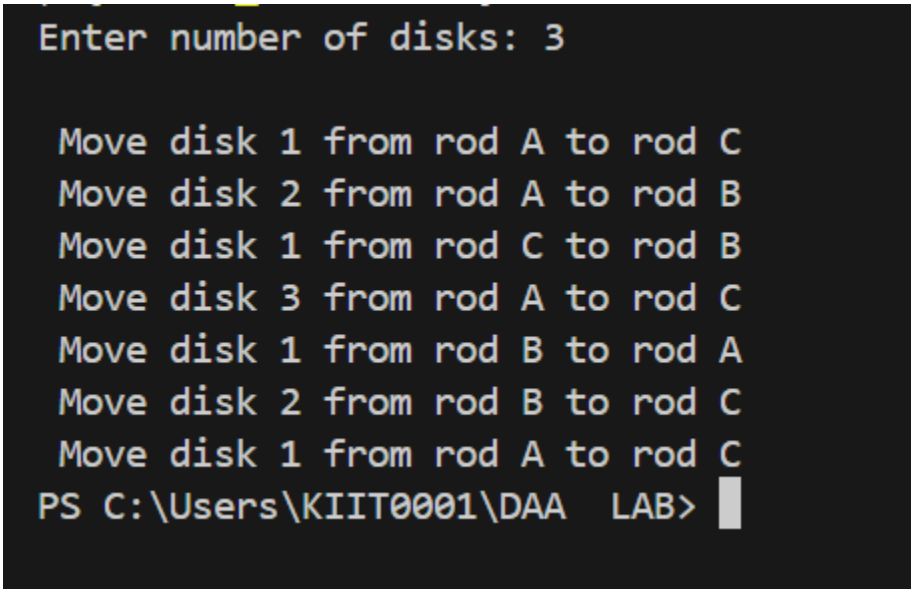
Recursive Program

```c
#include <stdio.h>

// Tower of Hanoi program in C using Recursion
void toH(int n, char rodA, char rodC, char rodB)
{
    if (n == 1)
    {
        printf("\n Move disk 1 from rod %c to rod %c",rodA ,rodC );
        return;
    }
    toH(n-1, rodA, rodB, rodC);
    printf("\n Move disk %d from rod %c to rod %c", n, rodA, rodC);
    toH(n-1, rodB, rodC,rodA);
}

int main()
{
    int no_of_disks ;
    printf("Enter number of disks: ");
    scanf("%d", &no_of_disks);
    toH(no_of_disks, 'A','C','B');
    return 0;
}
```

OUTPUT



Iterative Program

```c
#include <stdio.h>
#include <stdlib.h>

#define MAX 64   // maximum disks supported

// rods: 0 = Source (S), 1 = Auxiliary (A), 2 = Destination (D)
char rod[] = {'S', 'A', 'D'};
int stacks[3][MAX];  // each rod is a stack
int top[3];          // top index for each rod

// push operation
void push(int rodIndex, int disk) {
    stacks[rodIndex][++top[rodIndex]] = disk;
}

// pop operation
int pop(int rodIndex) {
    if (top[rodIndex] == -1) return -1;
    return stacks[rodIndex][top[rodIndex]--];
}
```

```c
// get top disk without popping
int peek(int rodIndex) {
    if (top[rodIndex] == -1) return 99999;  // big number if empty
    return stacks[rodIndex][top[rodIndex]];
}

// move disk between rods
void moveDisk(int a, int b) {
    if (top[b] == -1 || (top[a] != -1 && peek(a) < peek(b))) {
        printf("Move disk %d from rod %c to rod %c\n", peek(a), rod[a], rod[b]);
        push(b, pop(a));
    } else {
        printf("Move disk %d from rod %c to rod %c\n", peek(b), rod[b], rod[a]);
        push(a, pop(b));
    }
}

// iterative tower of hanoi
void towerOfHanoi(int n) {
    printf("Tower of Hanoi for %d disks:\n", n);

    int src = 0, aux = 1, dest = 2;

    // initialize tops
    for (int i = 0; i < 3; i++) top[i] = -1;

    // put disks in source stack
    for (int i = n; i > 0; i--)
        push(src, i);

    int totalMoves = (1 << n) - 1;

    if (n % 2 == 0) {
        int temp = aux;
        aux = dest;
        dest = temp;
    }

    for (int i = 1; i <= totalMoves; i++) {
        if (i % 3 == 0)
            moveDisk(aux, dest);
        else if (i % 3 == 1)
            moveDisk(src, dest);
```

```
        else
            moveDisk(src, aux);
    }
}

int main() {
    int n = 3; // number of disks
    towerOfHanoi(n);
    return 0;
}
```

OUTPUT

```
Tower of Hanoi for 3 disks:
Move disk 1 from rod S to rod D
Move disk 2 from rod S to rod A
Move disk 1 from rod D to rod A
Move disk 3 from rod S to rod D
Move disk 1 from rod A to rod S
Move disk 2 from rod A to rod D
Move disk 1 from rod S to rod D
PS C:\Users\KIIT0001\DAA  LAB>
```

2. Implementation of stack

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

struct Node {
    int data;
    struct Node* next;
};

int main() {
    struct Node* head = NULL;
    struct Node* temp;
    int n, value;
printf("enter number of elements: ");
      scanf("%d",&n);
      for(int i=0;i<n;i++){
```

```
            printf("Enter value: ");
            scanf("%d", &value);
            temp = (struct Node*)malloc(sizeof(struct Node));
            temp->data = value;
            temp->next = head;
            head = temp;
        }

            if (head == NULL) {
                printf("Stack is empty\n");
            } else {
                temp = head;
                head = head->next;
                printf("Removed: %d\n", temp->data);
                free(temp);
            }









    return 0;
}
```

OUTPUT

```
Enter number of elements: 5
Enter value: 2
Enter value: 1
Enter value: 34
Enter value: 67
Enter value: 78
Removed: 78
PS C:\Users\KIIT0001\DAA  LAB>
```