**5a. Develop a Program in C for the following Stack Applications. Evaluation of Suffix expression with single digit operands and operators: +, -, \*, /, %,^**

#include<stdio.h>

#include<stdlib.h>

#include<math.h>

int i, top = -1;

int op1, op2, res, s[20];

char postfix[90], symb;

void push(int item)

{

            top = top+1;

            s[top] = item;

}

int pop()

{

            int item;

            item  =  s[top];

            top = top-1;

            return item;

}

void main()

{

            printf("\nEnter a valid postfix expression:\n");

            scanf("%s", postfix);

            for(i=0; postfix[i]!='\0'; i++)

            {

                        symb = postfix[i];

                        if(isdigit(symb))

                        {

                                    push(symb - '0');

                        }

                        else

                        {

                                    op2 = pop();

                                    op1 = pop();

                                    switch(symb)

                                    {

                                                case '+':            push(op1+op2);

                                                                        break;

                                                case '-':             push(op1-op2);

                                                                        break;

case '\*':            push(op1\*op2);

                                                                        break;

                                                case '/':             push(op1/op2);

                                                                        break;

                                                case '%':           push(op1%op2);

                                                                        break;

                                                case '$':

                                                case '^':            push(pow(op1, op2));

                                                                        break;

                                                default :   push(0);

                                    }

                        }

            }

            res = pop();

            printf("\n Result = %d", res);

}

***Output:***

**To compile in Linux: gcc –lm 5.c**

Enter a valid postfix expression:

**623+-382/+\*2$3+**

**Result = 52**

Enter a valid postfix expression:

**42$3\*3-84/11+/+**

**Result = 46**

**5b. Solving Tower of Hanoi problem with n disks.**

#include<stdio.h>

#include<math.h>

**void tower(int n, char from\_peg,  char aux\_peg, char to\_peg);**

void main()

{

                int n;

                printf("\nEnter the number of disks: ");

                scanf("%d", &n);

**tower(n, 'A', 'B', 'C')**;                                         ***// A-> from\_peg B->aux\_peg C->to\_peg***

    printf("\nTotal number of moves = %0.0f", pow(2,n)-1 );

}

void tower(int n, char from\_peg,  char aux\_peg, char to\_peg)

***// A-> from\_peg B->aux\_peg C->to\_peg***

{

                if(n == 1)

                {

                                printf("\nMove disk %d from %c peg to %c peg", n,  from\_peg,  to\_peg);

                                return;

                }

***// move n-1 disks from A(from\_peg) to B(to\_peg) using C(aux\_peg)*** ***as auxiliary***

**tower(n-1, from\_peg,  to\_peg,  aux\_peg);**

                printf("\nMove disk %d from peg %c to %c peg", n, from\_peg,  to\_peg);

***// move n-1 disks from B(aux\_peg) to C(to\_peg) using A(from\_peg) as auxiliary***

**tower(n-1, aux\_peg, from\_peg, to\_peg);**

}

***Output:***

Enter the number of disks: **3**

**Move disk 1 from A peg to C peg**

**Move disk 2 from peg A to B peg**

**Move disk 1 from C peg to B peg**

**Move disk 3 from peg A to C peg**

**Move disk 1 from B peg to A peg**

**Move disk 2 from peg B to C peg**

**Move disk 1 from A peg to C peg**

Total number of moves  = **7**