ANVITA KUMAR
Roll No.: 2104097

```
//WAP to evaluate postfix expression using stack ADT

#include <stdio.h>

#include <string.h>

#include <ctype.h>

#include <stdlib.h>

#include <math.h>

struct Stack

{

 int top;

 unsigned capacity;

 int* array;

};

struct Stack* createStack( unsigned capacity )

{

 struct Stack* stack = (struct Stack*) malloc(sizeof(struct Stack));

 if (!stack) return NULL;

 stack->top = -1;

 stack->capacity = capacity;

 stack->array = (int*) malloc(stack->capacity * sizeof(int));

 if (!stack->array) return NULL;

 return stack;

}

int isEmpty(struct Stack* stack)

{

 return stack->top == -1 ;

}

char peek(struct Stack* stack)

{

 return stack->array[stack->top];
}
char pop(struct Stack* stack)
{
 if (!isEmpty(stack))
  return stack->array[stack->top--] ;
 return '$';
}
void push(struct Stack* stack, char op)
{
 stack->array[++stack->top] = op;
}
int isOperand(char ch)
{
    return (ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z') || (ch >= '0' && ch <= '9');
}
int Prec(char ch)
{
    switch (ch) {
    case '+':
    case '-':
        return 1;
    case '*':
    case '/':
        return 2;
    case '^':
        return 3;
    }
    return -1;
}
 char* infixToPostfix(char* exp)
```

ANVITA KUMAR
Roll No.: 2104097

```
{
int i, k;
   struct Stack* stack = createStack(strlen(exp));
   if(!stack)
      return NULL ;
   for (i = 0, k = -1; exp[i]; ++i) {
      if (isOperand(exp[i]))
         exp[++k] = exp[i];
      else if (exp[i] == '(')
         push(stack, exp[i]);
      else if (exp[i] == ')')  {
         while (peek(stack) != '(')
            exp[++k] = pop(stack);
         pop(stack);
      }
      else {
         while (!isEmpty(stack) && Prec(exp[i]) <= Prec(peek(stack)) && exp[i] != '^')
            exp[++k] = pop(stack);
         push(stack, exp[i]);
      }
   }
   while (!isEmpty(stack))
      exp[++k] = pop(stack);
   exp[++k] = '\0';
   printf("Resultant postfix expression: %s\n", exp);
   return exp;
}
int evaluatePostfix(char* exp)
{
   struct Stack* stack = createStack(strlen(exp));
```

ANVITA KUMAR
Roll No.: 2104097

```
 int i;
  if (!stack) return -1;
    printf("Token\t\tStack\n");
    for (i = 0; exp[i]; ++i) {
        if (isdigit(exp[i]))
            push(stack, exp[i] - '0');
        else
        {
            int val1 = pop(stack);
            int val2 = pop(stack);
            switch (exp[i]) {
            case '+': push(stack, val2 + val1); break;
            case '-': push(stack, val2 - val1); break;
            case '*': push(stack, val2 * val1); break;
            case '/': push(stack, val2/val1); break;
            case '^': push(stack, pow(val2, val1)); break;
            }
        }
        printf("%-16c", exp[i]);
        for (int i = 0; i <= stack->top; i++) {
            printf("%d ", stack->array[i]);
        }
        printf("\n");
    }
    return pop(stack);
}
int main()
{
    int c;
```

ANVITA KUMAR
Roll No.: 2104097

```
here:
    printf("You can enter infix or postfix expression, choose an option\n1. Infix expression\n2. Postfix
Expression\n");
scanf("%d", &c);
    char exp[20];
    switch(c) {
        case 1:
            printf("Enter the infix expression : ");
            scanf("%s", exp);
            printf ("infix evaluation: %d", evaluatePostfix(infixToPostfix(exp)));
            break;
        case 2:
            printf("Enter the postfix expression : ");
            scanf("%s", exp);
            printf ("postfix evaluation: %d", evaluatePostfix(exp));
            break;
        default:
            goto here;
    }
 return 0;
}
```