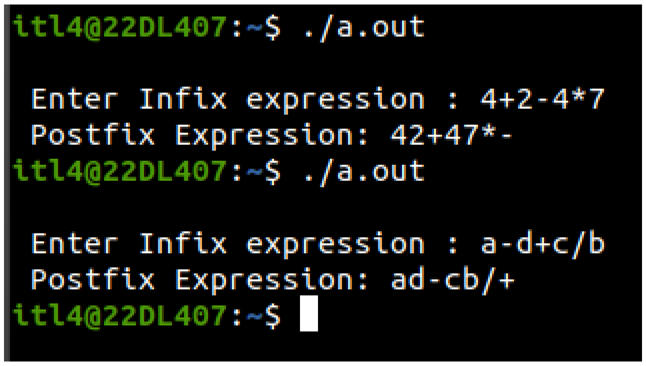
```
#include<stdio.h>
#include<stdlib.h>
#include<ctype.h>
#include<string.h>

#define SIZE 100
char stack[SIZE];
int top = -1;

/* == push operation == */
void push(char item)
{
    if(top >= SIZE-1)
    {
        printf("\n Stack Overflow.");
    }
    else
    {
        top = top+1;
        stack[top] = item;
    }
}

/* ==  pop operation  == */
char pop()
{
    char item ;

    if(top <0)
    {
        printf("stack under flow: invalid infix expression");
        getchar();
        /* underflow may occur for invalid expression */
        /* where ( and ) are not matched */
        exit(1);
    }
    else
    {
        item = stack[top];
        top = top-1;
        return(item);
    }
}
```

```
/* === define function that is used to determine whether any symbol is operator or not */

int is_operator(char symbol)
{
   if(symbol == '^' || symbol == '*' || symbol == '/' || symbol == '+' || symbol =='-')
   {
      return 1;
   }
   else
   {
   return 0;
   }
}

/* === define fucntion that is used to assign precendence to operator. */

int precedence(char symbol)
{
   if(symbol == '^')
   {
      return(3);
   }
   else if(symbol == '*' || symbol == '/')
   {
      return(2);
   }
   else if(symbol == '+' || symbol == '-')
   {
      return(1);
   }
   else
   {
      return(0);
   }
}

void InfixToPostfix(char infix_exp[], char postfix_exp[])
{
   int i, j;
   char item;
   char x;

   push('(');                /* push '(' onto stack */
   strcat(infix_exp,")");      /* add ')' to infix expression */
```

```
i=0;
j=0;
item=infix_exp[i];

while(item != '\0')
{
   if(item == '(')
   {
      push(item);
   }
   else if( isdigit(item) || isalpha(item))
   {
      postfix_exp[j] = item;          /* add operand symbol to postfix expr */
      j++;
   }
   else if(is_operator(item) == 1)     /* means symbol is operator */
   {
      x=pop();
      while(is_operator(x) == 1 && precedence(x)>= precedence(item))
      {
         postfix_exp[j] = x;         /* so pop all higher precendence operator and */
         j++;
         x = pop();                  /* add them to postfix expresion */
      }
      push(x);

      push(item);                    /* push current oprerator symbol onto stack */
   }
   else if(item == ')')              /* if current symbol is ')' then */
   {
      x = pop();                     /* pop and keep popping until */
      while(x != '(')                /* '(' encounterd */
      {
         postfix_exp[j] = x;
         j++;
         x = pop();
      }
   }
   else
   {  /* if current symbol is neither operand not '(' nor ')' and nor operator */
      printf("\nInvalid infix Expression.\n");
      getchar();
      exit(1);
```

```
        }
        i++;


        item = infix_exp[i];
    }
    if(top>0)
    {
        printf("\nInvalid infix Expression.\n");
        getchar();
        exit(1);
    }

      postfix_exp[j] = '\0'; /* add sentinel else puts() fucntion */



    }

/* === main function begins === */
int main()
{
    char infix[SIZE], postfix[SIZE];

    printf("\n Enter Infix expression : ");
    gets (infix);

    InfixToPostfix(infix,postfix);
    printf(" Postfix Expression: ");
    puts(postfix);

    return 0;
}
```



```
itl4@22DL407:~$ ./a.out

 Enter Infix expression : 4+2-4*7
 Postfix Expression: 42+47*-
itl4@22DL407:~$ ./a.out

 Enter Infix expression : a-d+c/b
 Postfix Expression: ad-cb/+
itl4@22DL407:~$ 
```

Name : Yash Bid
Roll no : 05
Exp 3