NAME: AGNIV PRAMANICK **SECTION:** A **USN:** 1NT21IS017 **DATE:** 29/12/22

Q. Write a program to Convert POSTFIX to PREFIX

THEORY

Prefix Expression:

Another way to describe anything is with a prefix notation, which does not require knowledge about precedence or associativity but does when used with an infix notation. It is also known as **polish notation**. In prefix notation, an operator comes before the operands.

The syntax of prefix notation is given below:
<operator> <operand> <operand>

Example:

A B - C D - /

Postfix Expression:

When the operator is written after the operands, then it is known as **postfix notation**. Operand does not have to be always a constant or a variable; it can also be an expression itself.

The syntax for postfix notation is given below:

<operand> <operand> <operator>

Example:
/ - A B - C D

*+AB-CD is the obtained PREFIX expression.

ALGORITHM

```
 Step-1: Scan the postfix expression from left to right.
Step-2: If the element is an operand, then push it into the stack.
Step-3: If the element is an operator, then pop two operands from the stack.
Step-4: Create an expression by concatenating two operands and adding operator before the operands.
Step-5: Push the result back to the stack.
Step-6: Repeat the above steps until we reach the end of the postfix expression.
```

CODE

```
#include<stdio.h>

#include<string.h>        //library functions are included

#include<stdlib.h>

#define MAX 20   // max size is 20
```

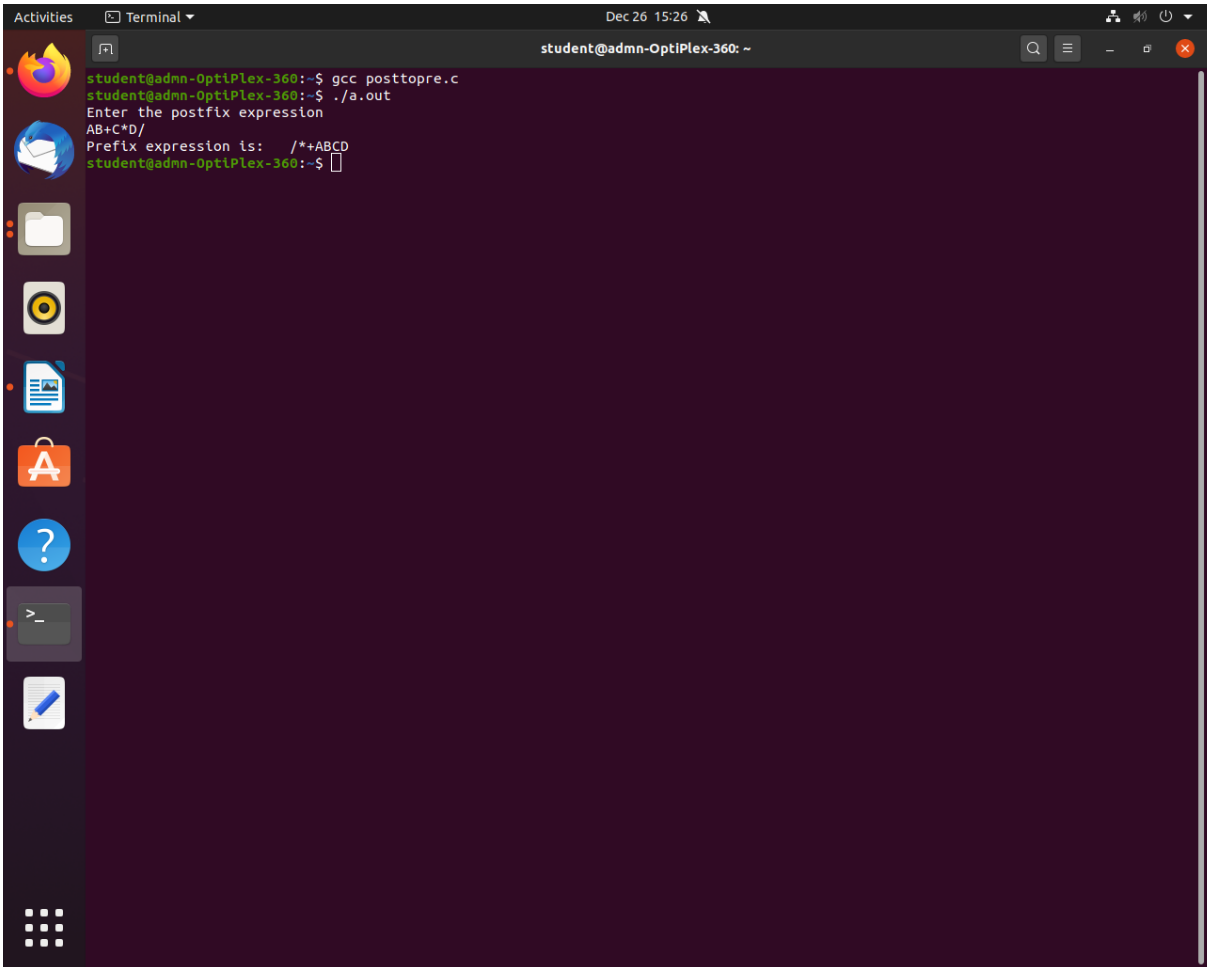
```
char str[MAX], stack[MAX];          a string of max size is taken of chartype
int top = -1;
 //-----------------------------------------------------------------------------------------
void push (char c)     // push operation is defined
{
  stack[++top] = c;       //top incremented
}
 //-----------------------------------------------------------------------------------------
char pop()          //pop function is defined
{
  return stack[top--];        //top decremented
}
//------------------------------------------------------------------------------------------

// A utility function to check if the given character is operand
int checkIfOperand(char ch)
{
  return (ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z');  //condition given
}

//------------------------------------------------------------------------------------------
//function to check if it is an operator
int isOperator(char n)    //character is given to check operator
{
  switch (n) {
  case '+':  //addition
  case '-':    //difference
  case '/':      //multiplication
  case '*':   //division
    return 1;
  }
  return 0;
}
 //------------------------------------------------------------------------------------
void postfixToprefix()        //conversion will take place here
{
  int n, i, j = 0;
  char c[20];                          //character type array
  char a, b, op;
  printf("Enter the postfix expression\n");
  scanf("%s", str);
  n = strlen(str);                         //string length is taken
  for (i = 0; i < MAX; i++)
    stack[i] = '\0';                           // means null equivalent
  printf("Prefix expression is:\t");
  for (i = n - 1; i >= 0; i--)                   //condition
  {
    if (isOperator(str[i]))            //operator is checked
    {
      push(str[i]);                //string is pushed
    } else
    {
      c[j++] = str[i];
      while ((top != -1) && (stack[top] == '#'))           //condition
      {
        a = pop();              //pop function called
        c[j++] = pop();
      }
      push('#');
    }
  }
  c[j] = '\0';
  i = 0;
  j = strlen(c) - 1;                j value is equivalent to string length of c -1
  char d[20];
  while (c[i] != '\0') {
    d[j--] = c[i++];
  }
  printf("%s\n", d);        //print the value
}
//------------------------------------------------------------------------------------------
int main()            //main function defined
{
  postfixToprefix();          //conversion function is called
  return 0;
}
//end of code
//------------------------------------------------------------------------------------------
```

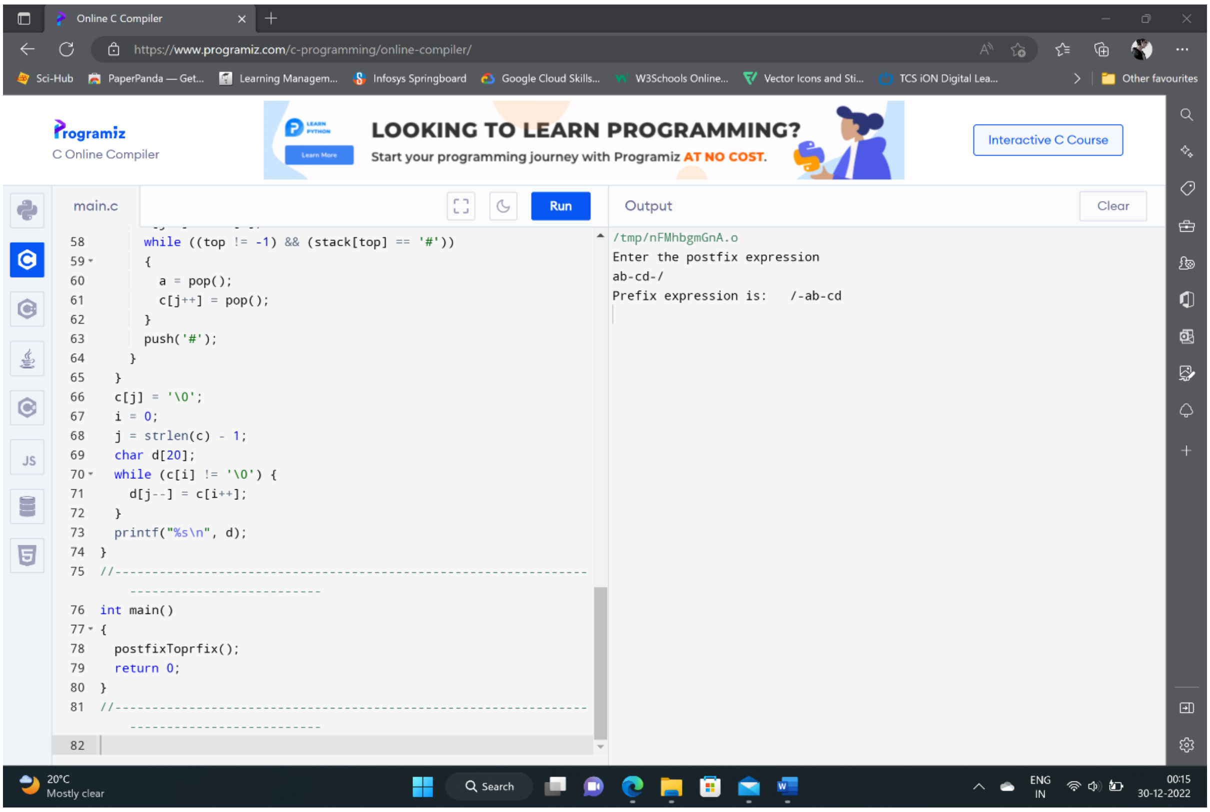
OUTPUT

NOTE: BELOW FIRST ONE IS A LINUX O/P SCREENSHOT



```
student@admn-OptiPlex-360:~$ gcc posttopre.c
student@admn-OptiPlex-360:~$ ./a.out
Enter the postfix expression
AB+C*D/
Prefix expression is:   /*+ABCD
student@admn-OptiPlex-360:~$ 
```

THE NEXT SCREENSHOT IS FROM WINDOWS ON AN ONLINE C COMPILER (programiz.com)



```
      while ((top != -1) && (stack[top] == '#'))
      {
        a = pop();
        c[j++] = pop();
      }
      push('#');
    }
  }
  c[j] = '\0';
  i = 0;
  j = strlen(c) - 1;
  char d[20];
  while (c[i] != '\0') {
    d[j--] = c[i++];
  }
  printf("%s\n", d);
}
//-------------------------------------------------------------------
    -------------------------
int main()
{
  postfixToprfix();
  return 0;
}
//-------------------------------------------------------------------
    -------------------------

```

```
/tmp/nFMhbgmGnA.o
Enter the postfix expression
ab-cd-/
Prefix expression is:   /-ab-cd
```