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

Q. Write a program to convert from infix to postfix expression.

THEORY

WHAT IS INFIX EXPRESSION?

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

For example,

(A - B) / (C - D)

In the above expression, both the expressions of the multiplication operator are the operands, i.e., **(A - B)**, and **(C - D)** are the operands.

syntax

<operand> <operator> <operand>

WHAT IS POSTFIX EXPRESSION?

The postfix expression is an expression in which the operator is written after the operands. For example, the postfix expression of infix notation (2+3) can be written as 23+.

Some key points regarding the postfix expression are:

- In postfix expression, operations are performed in the order in which they have written from left to right.
- It does not any require any parenthesis.
- We do not need to apply operator precedence rules and associativity rules.

ALGORITHM

Step1: If the input character is an operand, print it.
Step2: If the input character is an operator-

- If stack is empty, push it to the stack.
- If its precedence value is greater than the precedence value of the character on top, push.
- If its precedence value is lower or equal then pop from stack and print while precedence of top char is more than the precedence value of the input character.

Step3: If the input character is ‘)’, then pop and print until top is ‘(‘. (Pop ‘(‘but don’t print it.)
Step4: If stack becomes empty before encountering ‘(‘, then it’s an invalid expression.
Step5: Repeat steps 1-4 until input expression is completely read.
Step6: Pop the remaining elements from stack and print them.

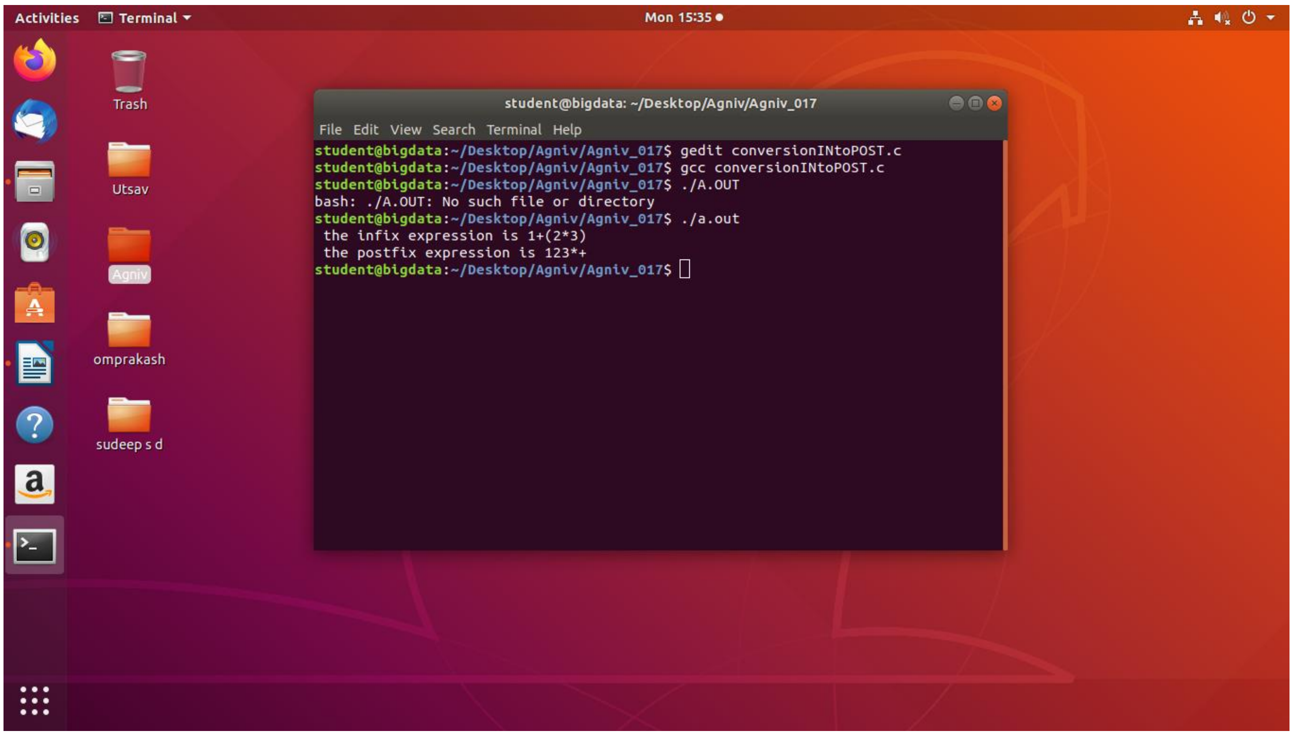
CODE

```
// to convert an infix expression to a postfix expression

#include<stdio.h>
#include<string.h>

//char stack
char stack[25]; //declare an array called stack with 25 as the size
int top=-1;     // declare a pointer top and value should be -1
```

```
//now let us declare a user defined function to push our value
void push(char item)
{
 stack[++top]=item;
}

//now let us declare a user defined function to pop our value
char pop()
{
 return stack[top--];
}

//list of precedence
int precedence(char symbol)
{
 switch(symbol) //create a switch case for different symbols applicable in this program
 {
   case '+':
   case '-':
           return 2;
           break;
   case '*':
   case '/':
           return 3;
           break;
   case '^':
           return 4; //most precedence is given as return value is highest
           break;
   case '(':
   case ')':
   case '#':
           return 1; //least precedence is given as return value is smallest
           break;
 }
}

//to check the symbol given by user
int isoperator(char symbol){
switch(symbol){
   case '+':
   case '-':
   case '*':        // all these symbols are valid, anything else will return 0
   case '/':
   case '(':
   case ')':
          return 1;
          break;
   default:
          return 0;
 }
}
```

```
//convert from infix to postfix
void convert(char infix[], char postfix[])
{
 int i,j=0,symbol;  //i,j,symbol is initialised
 stack[++top]='#';  //increment the top value
 for(i=0;i<strlen(infix);i++)   //for condition with respect to value of i
 {
  symbol=infix[i];
  if(isoperator(symbol)==0)  //condition is given that isoperator(symbol) should be zero
  {
   postfix[j]=symbol;
   j++;    // increment the j
  }
  else
  {
   if(symbol=='(')  //satisfies if the condition is true that is symbol is (
   {
    push(symbol); //call the push function
   }
   else
   {
    if(symbol==')')  //satisfies if the condition is true that is symbol is )
    {
     while(stack[top]!='(') //loop will run for the conditon that stack[top] is not equal to (
     {
      postfix[j]=pop();
      j++;  // j is incremented
     }
     pop();  //call the pop function
    }

    else
    {
     if(precedence(symbol)>precedence(stack[top]))  //condititon to be satisfied for calling push
function
     {
      push(symbol);  //call the push function
     }

     else
     {
      while(precedence(symbol)<=precedence(stack[top]))  //condititon to be satisfied to run the
while loop
      {
       postfix[j]=pop();  //pop function is called
       j++;  // j is incremented
      }
      push(symbol);
     }
    }
   }
  }
 }

 while(stack[top]!='#') //to run the loop array should not be equal to #
 {
  postfix[j]=pop(); //pop function is called
  j++; //j value incremented
 }
 postfix[j]='\0';
}

//run the main function
void main()
{
 char infix[50]="1+(2*3)", postfix[30],symbol; //infix is given by the user
 convert(infix,postfix); // call the convert function
 printf(" the infix expression is %s \n",infix); //print the infix expression
 printf(" the postfix expression is %s \n",postfix); //print the postfix expression
}
```

SCREENSHOT OF THE OUTPUT


Activities
Terminal
Mon 15:35
Trash
Utsav
Agniv
omprakash
sudeep s d
student@bigdata: ~/Desktop/Agniv/Agniv_017
File Edit View Search Terminal Help

```
student@bigdata:~/Desktop/Agniv/Agniv_017$ gedit conversionINtoPOST.c
student@bigdata:~/Desktop/Agniv/Agniv_017$ gcc conversionINtoPOST.c
student@bigdata:~/Desktop/Agniv/Agniv_017$ ./A.OUT
bash: ./A.OUT: No such file or directory
student@bigdata:~/Desktop/Agniv/Agniv_017$ ./a.out
 the infix expression is 1+(2*3)
 the postfix expression is 123*+
student@bigdata:~/Desktop/Agniv/Agniv_017$
```