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

Q. To write a program so as to evaluate the postfix expression.

THEORY

The Postfix notation is used to represent algebraic expressions. The expressions written in postfix form are evaluated faster compared to infix notation as parenthesis are not required in postfix. We have discussed infix to postfix conversion. Here, evaluation of postfix expressions is explained.

- Scan the expression from left to right.
- If we encounter any operand in the expression, then we push the operand in the stack.
- When we encounter any operator in the expression, then we pop the corresponding operands from the stack.
- When we finish with the scanning of the expression, the final value remains in the stack.

ALGORITHM

```
Step 1 − scan the expression from left to right
Step 2 − if it is an operand push it to stack
Step 3 − if it is an operator pull operand from stack and perform operation
Step 4 − store the output of step 3, back to stack
Step 5 − scan the expression until all operands are consumed
Step 6 − pop the stack and perform operation
```

CODE

```
//to evaluate the postfix expression after converting it from infix
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<ctype.h>


//char stack
char stack[25];  //create a stack array of 25 size
int top=-1; // value of -1 is given to top

// creating a user defined function to push elements
void push(char item)
{
 stack[++top]=item; // incrementing top and equating thae array to item
}

// creating a user defined function to pop elements
int pop()
{
 return stack[top--];
}
```

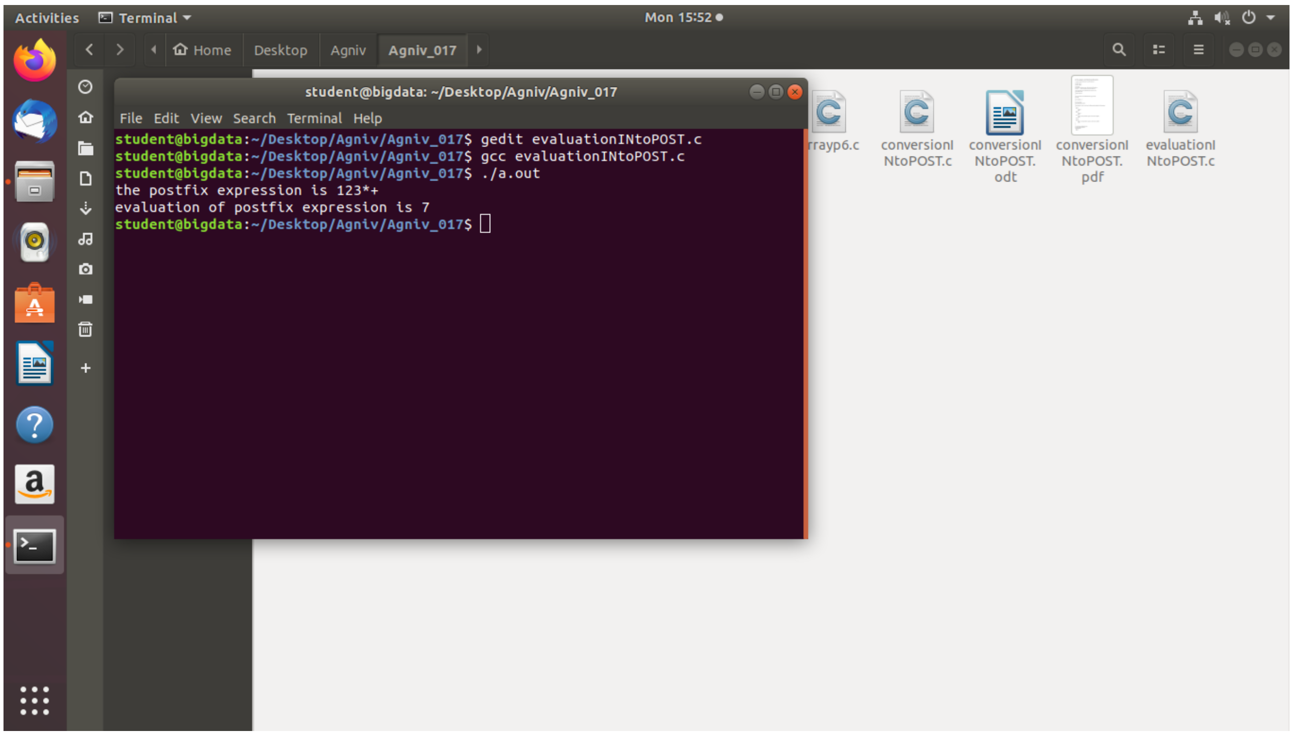
```
//evaluation of POSTFIX
// creating a user defined function to evaluate elements
int evaluate(char *postfix)
 {
  char ch; // declaring ch as a character
  int op1,op2; //declaring two operands
  int i=0;  //giving i a value of 0
  while((ch=postfix[i++])!='\0') //while loop with the condition, ch is not equal to null character
  {
   if(isdigit(ch)) //if condition implimented
   {
    push(ch-'0'); // to get the ASCII value from ch and calling push function
   }
   else
   {
    op2=pop(); // operand 2 is equal to pop function
    op1=pop(); // operand 2 is equal to pop function
   }

   switch(ch) // creating a switch case for the following cases
   {
    case'+': push(op1+op2); //addition
            break;
    case'-': push(op1-op2); //subtraction
            break;
    case'*': push(op1*op2); //multiplication
            break;
    case'/': push(op1/op2); //division
            break;
    case'^': push(op1^op2); //power
            break;

    }
   }
   return stack[top];        // return the stack array value
  }

//now writing the main function
void main()
{
char postfix[30]="123*+"; //giving postfix the expression
printf("the postfix expression is %s \n",postfix); //printing the postfix expression.
printf("evaluation of postfix expression is %d \n",evaluate(postfix)); //printing the evaluated answer
}
//code is ended
```

SCREENSHOT OF THE OUTPUT.



```
student@bigdata:~/Desktop/Agniv/Agniv_017$ gedit evaluationINtoPOST.c
student@bigdata:~/Desktop/Agniv/Agniv_017$ gcc evaluationINtoPOST.c
student@bigdata:~/Desktop/Agniv/Agniv_017$ ./a.out
the postfix expression is 123*+
evaluation of postfix expression is 7
student@bigdata:~/Desktop/Agniv/Agniv_017$ 
```