Jawahar Education Societys Annasaheb Chudaman Patil College of Engineering, Kharghar, Navi Mumbai

NAME: PRIYUSH BHIMRAO KHOBRAGADE

PRN NO: 211112018

SUBJECT: DATA STRUCTURES LAB

AIM: Convert an Infix expression to Postfix expression using stack ADT

PAGE NO.:
DATE.: / / 20

Practical No:- 02

Aim :- Convert an Infix expression to postfix expression using Stack ADT.

Theory :-

- **Infix expression**

"An infix expression is an expression in which operators (+, -, *, /) are written between the two operands."

Ex.

A+B

A+B-C

- **Postfix expression :-**

"In postfix expression, the operator is written after the operand. It also know as Reverse polish notation.

Ex. AB+

AB+C-

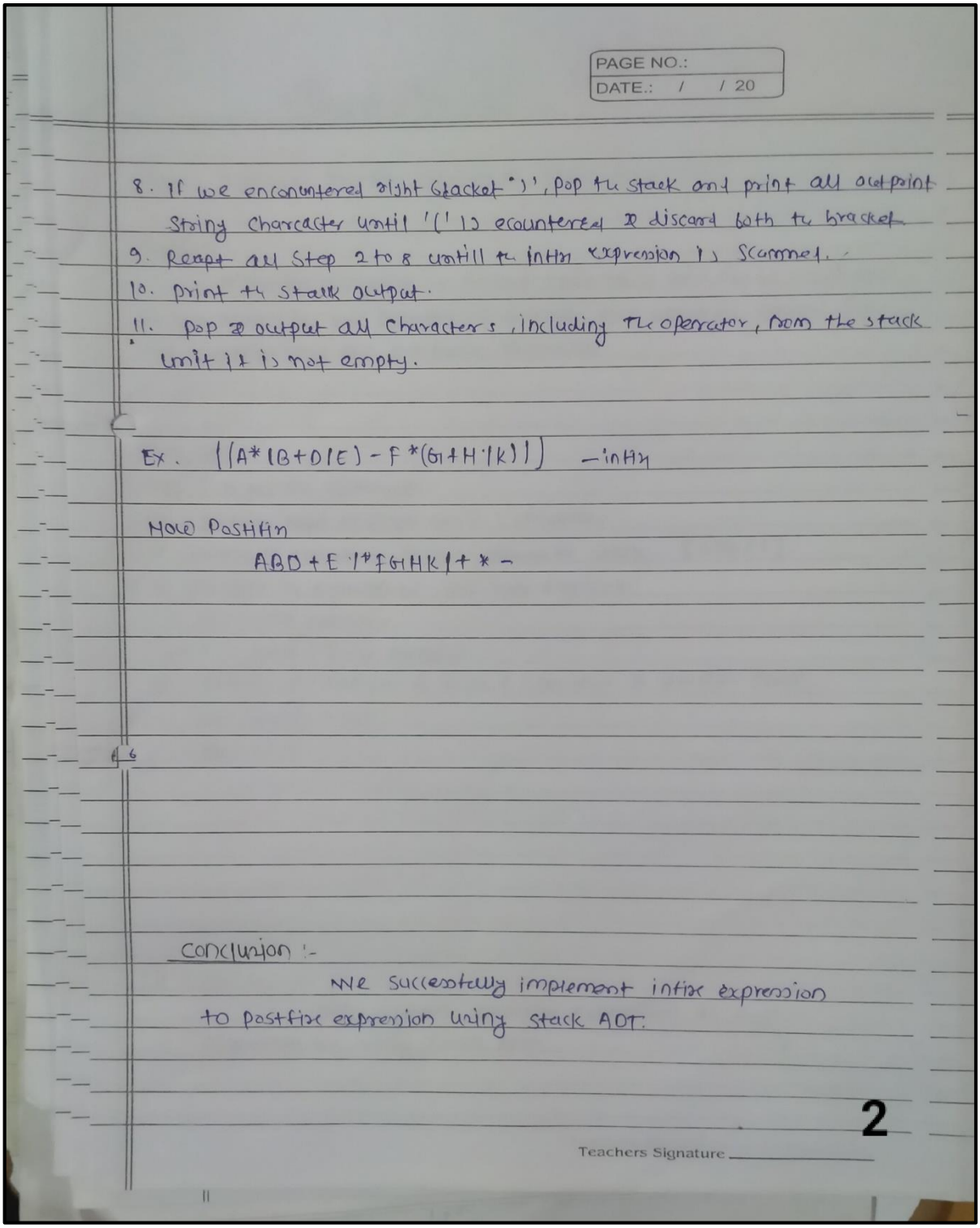
- **Algorithm :-**

1. Initialize the stack.
2. Scan the operand from Left to Right.
3. If the leftmost character is an operand, set output postfix string.
4. If stack is empty or contains the '(' symbol push the operand into stack.
5. If Scanned operator has Higher priedence then existing priedence operator in stack or if the stack is empty, put it on the stack.
6. If Scanned operator has lower priedence then existing priedence operator in stack, pop all the stack operator.
7. If scanned character is a leaf bracket '(', push it into the stack.

Teachers Signature _____

Priyush B. Khobragade

AIM: Convert an Infix expression to Postfix expression using stack ADT



PAGE NO.:
DATE: / / 20

8. If we encountered right bracket ')', pop the stack and print all output string character until '(' is encountered & discard both the bracket.
9. Repeat all step 2 to 8 until the infix expression is scanned.
10. print the stack output.
11. pop & output all characters, including the operator, from the stack until it is not empty.

Ex. [(A*(B+D/E)-F*(G+H/K))] -infix

Now Postfix

ABD+E/*FGHK/+*-

Conclusion :-

We successfully implement infix expression to postfix expression using stack ADT.

2

Teachers Signature _____

AIM: Convert an Infix expression to Postfix expression using stack ADT

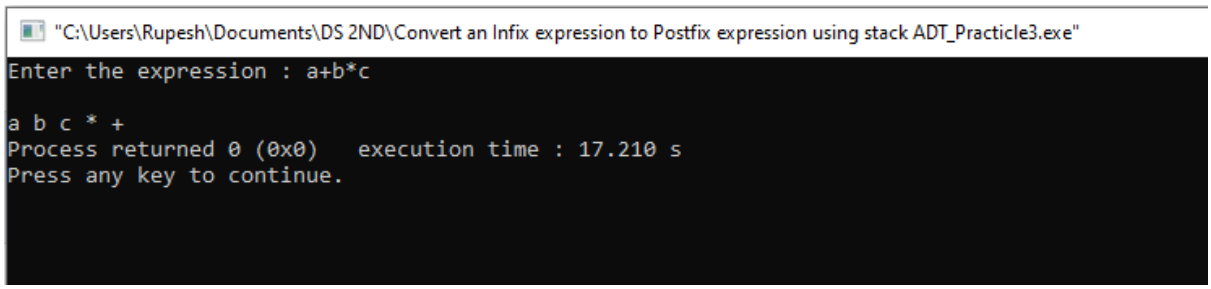
Input:-

```c
#include<stdio.h>
#include<conio.h>
#include<ctype.h> /*header file of the C Standard Library*/

char stack[100];
int top = -1;

void push(char x)   /*define push operation*/
{
   stack[++top] = x;
}

char pop()       /*define pop operation*/
{
   if(top == -1)
      return -1;
   else
      return stack[top--];
}

int priority(char x)   /*define priority*/
{
   if(x == '(')
      return 0;
   if(x == '+' || x == '-')
      return 1;
   if(x == '*' || x == '/')
      return 2;
   return 0;
}

int main()
{
   char exp[100];
   char *e, x;
   printf("Enter the expression : ");
   scanf("%s",exp);
   printf("\n");
   e = exp;

   while(*e != '\0')    /*define condition*/
   {
      if(isalnum(*e))
         printf("%c ",*e);
      else if(*e == '(')
         push(*e);
      else if(*e == ')')
      {
         while((x = pop()) != '(')
            printf("%c ", x);
      }
      else
      {
         while(priority(stack[top]) >= priority(*e))
            printf("%c ",pop());
         push(*e);
      }
      e++;
   }

   while(top != -1)
   {
      printf("%c ",pop());
   }return 0;
}
```

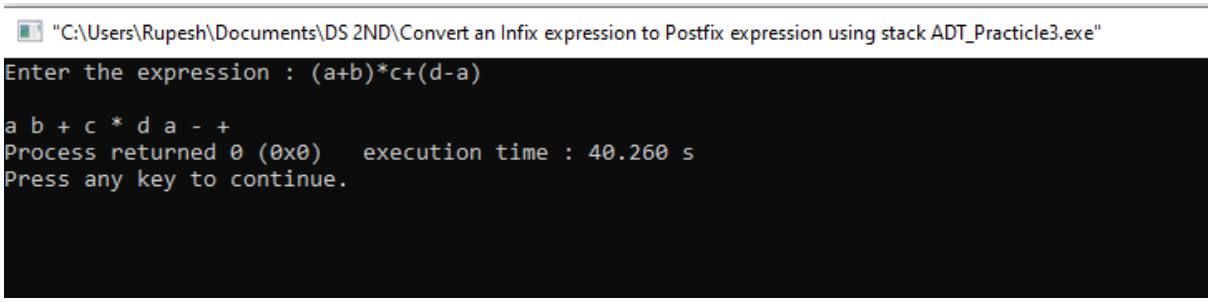
AIM: Convert an Infix expression to Postfix expression using stack ADT

Output:-

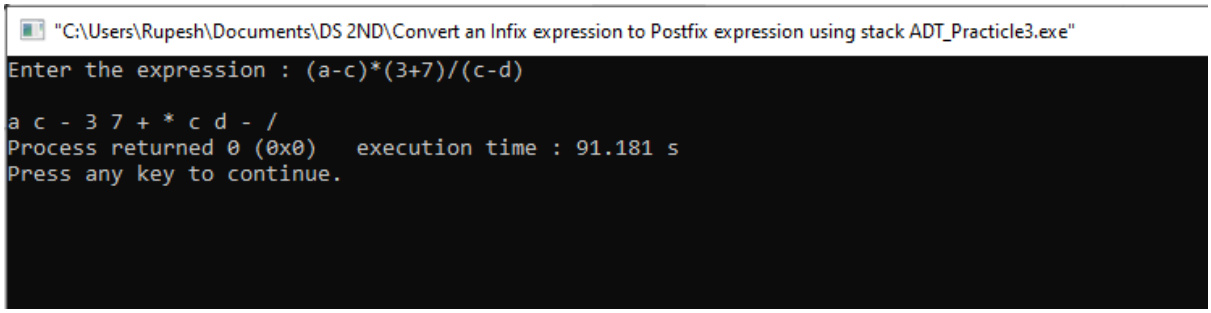
Test 01



Test 02



Test 03



Conclusion: - We successfully implement infix expression to postfix expression using stack ADT