Experiment:- 1.4

Write a program to calculate postfix expressin of A*B+C/D

Student Name: Neha Sharma **UID:- 20BCS4576**
Branch: CSE-IOT **Section/Group:- A**
Semester: 3rd **Date of Performance:- 13/09/2021**
Subject Name:- DS LAB **Subject Code:-20CSP-236**

1. Aim/Overview of the practical:- Write a program to calculate postfix expressin of A*B+C/D

2. Task to be done:-

- We define an empty stack at first. We then push to the end of X
- We scan the infix expression from left to right till the stack is empty
- We define a priority function to check for the order of precedence.
- In the main function, we ask the user to enter the required expression.
- Conversion of expression takes place by calling the previous functions.
- We add a few more functions.
- Result is displayed.

3. Algorithm/Flowchart:

Step 1: Start
Step 2: Start reading the infix expression from left to right.
Step 3: Repeat Step 4 to 7 for each element until the Stack is empty.
Step 4: If we scan a operand we output it, print it.
Step 5: Else,
Step 5.1: If the scanned operator is greater in precedence than the operator in the stack or if the stack is empty or the stack contains a "(", push it.
Step 5.2: Else, Pop all the operators having greater or equal precedence than that of the scanned operator. After doing that Push the scanned operator to the stack. In case there is a parenthesis while popping then stop and push the scanned operator in the stack.

Step 6: If a '(' is encountered, push it onto Stack.
Step 7: If a ')' is encountered, repeatedly pop from Stack and output it until a '(' is encountered
Step 8: The output is printed in postfix notation
Step 9: Stop

4. Steps for experiment/practical:

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

char stack[20];

int top = -1;

void push(char x)

{
```

```
  stack[++top] = x;
}


char pop()
{
  if(top == -1)
      return -1;
  else
      return stack[top--];
}
int priority(char x)
{
  if(x == '(')
      return 0;
  if(x == '+' || x == '-')
      return 1;
```

```
  if(x == '*' || x == '/')
     return 2;
}


main()
{
  char exp[20];
  char *e, x;
  printf("Enter the expression :: ");
  scanf("%s",exp);
  e = exp;
  while(*e != '\0')
  {
     if(isalnum(*e))
        printf("%c",*e);
     else if(*e == '(')
```

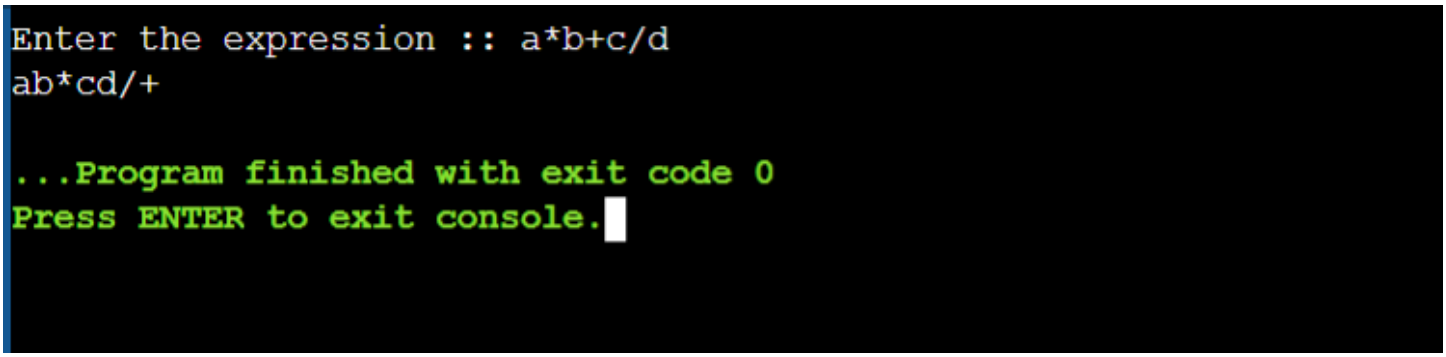
```
        push(*e);
    else if(*e == ')')
    {
        while((x = pop()) != '(')
            printf("%c", x);
    }
    else
    {
        while(priority(stack[top]) >= priority(*e))
            printf("%c",pop());
        push(*e);
    }
    e++;
}
while(top != -1)
{
```

```
    printf("%c",pop());

  }

}
```

5. <u>Output: Image of sample output to be attached here</u>



Learning outcomes (What I have learnt):

1. **Conversion of infix to postfix**

2. **Go to know about how precedence of operators works.**

Evaluation Grid (To be created as per the SOP and Assessment guidelines by the faculty):

Sr. No.	Parameters	Marks Obtained	Maximum Marks
1.			
2.			
3.			