Experiment: 2.3

Student Name: Rohit Panghal **UID: 21BCS9294**
Branch: CSE **Section/Group: 902-A**
Semester: 3rd **Date:**
Subject Name: Data Structures and Algorithms **Subject code: 21CSH-211**

AIM:

Write a program to demonstrate the use of stack [implemented using lineararray] in converting arithmetic expression from infix notation to postfix notation.

ALGORITHM:

Step 1: Start

Step 2: Scan the infix expression from left to right.

Step 3: Print the operand as they arrive.

Step 4: If the stack is empty or contains a left parenthesis on top, push the incomingoperator on to the stack.

Step 5: If the incoming symbol is '(', push it on to the stack.

Step 6: If the incoming symbol is ')', pop the stack and print the operators until the leftparenthesis is found.

Step 7: If the incoming symbol has higher precedence than the top of the stack, push iton the stack.

Step 8: If the incoming symbol has lower precedence than the top of the stack, pop andprint the top of the stack. Then test the incoming operator against the new top of the stack.

Step 9: If the incoming operator has the same precedence with the top of the stack thenuse the associativity rules. If the associativity is from left to right then pop and print the top of the stack then push the incoming operator. If the associativity is from right to leftthen push the incoming operator.

Step 10: At the end of the expression, pop and print all the operators of the stack.

Step 11: End.

PROGRAM CODE:

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

char stack[100];
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 == '(')
```

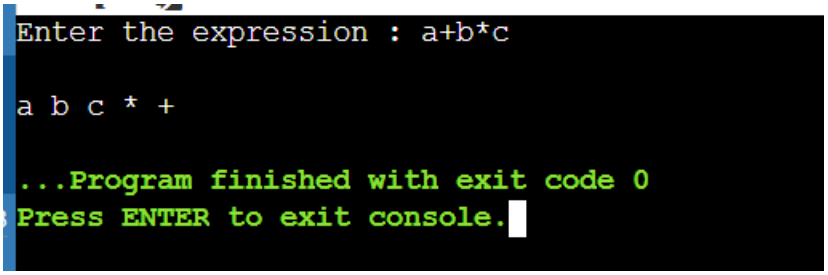
```
        return 0;
    if(x == '+' || x == '-')
        return 1;
    if(x == '*' || x == '/')
        return 2;
    return 0;
}

int main()
{
    char exp[100];
    char *e, x;
    printf("Enter the expression : ");
    scanf("%s",exp);
    printf("\n");
    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());
    }return 0;
}
```

OUTPUT:



LEARNING OUTCOME:

1. Learnt what is insertion sorting.
2. Learnt the algorithm of algorithm sorting.
3. Learnt how to code for insertion sort in C.