**Experiment 4a**

**Aim:**Implement postfix expression evaluation using stack.

**Algorithm:**

1. Initialise a stack and declare the push and pop functions.

2. If the input is a digit, push the input.

3. If the input is an operator, pop twice and push the result of the expression.

4. Return the result or continue the process until we reach the end of stack.

**Source Code:**

#include <stdio.h>

#include <ctype.h>

int st[10];

int top = -1;

void push(int x)

{

    st[++top] = x;

}

int pop()

{

    return st[top--];

}

int main()

{

    char express[10];

    char \*exp;

    int a, b, res, num;

    printf("Enter expression: ");

    scanf("%s", express);

    exp = express;

    while (\*exp != '\0')

    {

        if (isdigit(\*exp))

        {

            num = \*exp - 48;

            push(num);

        }

        else

        {

            a = pop();

            b = pop();

            switch (\*exp)

            {

            case '+':

            {

                res = a + b;

                break;

            }

            case '-':

            {

                res = a - b;

                break;

            }

            case '\*':

            {

                res = a \* b;

                break;

            }

            case '/':

            {

                res = a / b;

                break;

            }

            }

            push(res);

        }

        exp++;

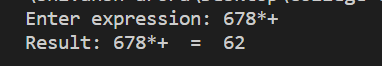
    }

    printf("Result: %s  =  %d\n", express, pop());

    return 0;

}

**Output:**



**Experiment 4b**

**Aim:**Convert infix expression to postfix expression using stack.

**Algorithm:**

1. Initialise a stack and declare the push and pop functions.

2. If the input is alphanumeric, push the input.

3. If the input is an operator, pop twice and push the result of the expression.

4. Continue the process until we reach the end of stack.

**Source Code:**

#include<stdio.h>

#include<ctype.h>

char stack[10];

int top = -1;

void push(char x)

{

    stack[++top] = x;

}

char pop()

{

    if(top == -1)

        return -1;

    else

        return stack[top--];

}

int order(char x)

{

    if(x == '(')

        return 0;

    if(x == '+' || x == '-')

        return 1;

    if(x == '\*' || x == '/')

        return 2;

    return 0;

}

int main()

{

    char express[10];

    char \*exp, x;

    printf("Expression: ");

    scanf("%s",express);

    exp = express;

    while(\*exp != '\0')

    {

        if(isalnum(\*exp))

            printf("%c ",\*exp);

        else if(\*exp == '(')

            push(\*exp);

        else if(\*exp == ')')

        {

            while((x = pop()) != '(')

                printf("%c ", x);

        }

        else

        {

            while(order(stack[top]) >=order(\*exp))

                printf("%c ",pop());

            push(\*exp);

        }

        exp++;

    }

    while(top != -1)

    {

        printf("%c ",pop());

    }return

    0;

}

**Output:**

