# **Name: Abdurrahman Qureshi**

# **Roll No: 242466**

Practical No: 9

**1)** **Demonstrate the** Implementation of infix to postfix conversion and evaluation of postfix conversion.

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)

    {

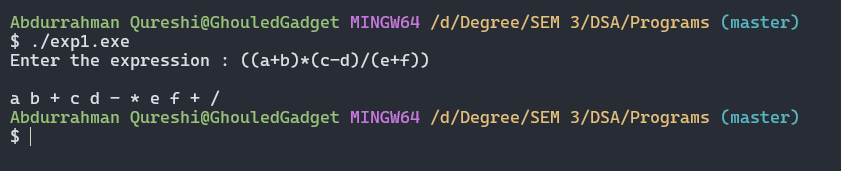
        printf("%c ", pop());

    }

    return 0;

}

OUTPUT



Tools used :

Software: Dev c++

Hardware: Lab Computers

References: Mam notes.

Conclusion

Infix to postfix conversion simplifies expressions by eliminating parentheses. Postfix evaluation uses a stack for efficient calculation. This method enhances computational efficiency, especially for evaluating complex expressions in calculators and interpreters.