

Institute of Computer Technology
B. Tech. Computer Science and Engineering
Sub: DS
Course Code: 2CSE302

Practical – 6

Name: Jaymin Gondaliya
Enrollment No: 23162171007
Sem - 3
Branch: CS
Class: A
Batch: 32

Problem Definition-1: Write a C/C++ program to convert infix notation to prefix notation using stack

Code:

```
#include <stdio.h>

#define max 100

int top = -1, a[max]; // Stack to hold operators with `top` initialized to -1
(indicating an empty stack)

// Function to push an operator onto the stack
void push(char x)
{
    a[++top] = x; // Increment `top` and then store `x` at that position
}

// Function to pop an operator from the stack
char pop()
{
    if (top == -1)
        return -1; // Return -1 if the stack is empty
    else
```

```
        return a[top--]; // Return the top element and then decrement `top`
    }

// Function to define precedence of operators
int prcd(char c)
{
    if (c == '(' || c == ')')
        return 0; // '(' and ')' have the lowest precedence
    else if (c == '+' || c == '-')
        return 1; // '+' and '-' have the same precedence
    else if (c == '*' || c == '/')
        return 2; // '*' and '/' have the same precedence, which is higher
    than '+' and '-'
}

// Function to reverse a string
void reverse(char exp[max])
{
    int length = strlen(exp);
    for (int i = 0; i < length / 2; i++)
    {
        char temp = exp[i];
        exp[i] = exp[length - i - 1];
        exp[length - i - 1] = temp;
    }
}

// Function to convert infix expression to prefix
void infixtoprefix(char infix[max], char prefix[max])
{
    char temp, x;
    int i = 0, j = 0;

    reverse(infix); // Reverse the infix expression

    while (infix[i] != '\0') // Loop through the entire reversed infix
    expression
    {
        temp = infix[i];

        if (isalnum(temp)) // If the character is an operand (number/letter)
        {
            prefix[j++] = temp; // Add it directly to the prefix expression
        }
        else if (temp == ')')
        {
            push(temp); // Push ')' onto the stack
        }
    }
}
```

```
        else if (temp == '(')
        {
            while ((x = pop()) != ')') // Pop from the stack until ')' is
found
            {
                prefix[j++] = x; // Add the popped operators to prefix
            }
        }
        else // If the character is an operator
        {
            while (prcd(a[top]) >= prcd(temp)) // Check precedence and pop
operators from stack if needed
            {
                prefix[j++] = pop(); // Add the popped operator to prefix
            }
            push(temp); // Push the current operator onto the stack
        }
        i++;
    }

    while (top != -1) // Pop any remaining operators in the stack
    {
        prefix[j++] = pop();
    }

    prefix[j] = '\0'; // Null-terminate the prefix expression

    reverse(prefix); // Reverse the prefix expression to get the correct
result
}

int main()
{
    char infix[max], prefix[max];

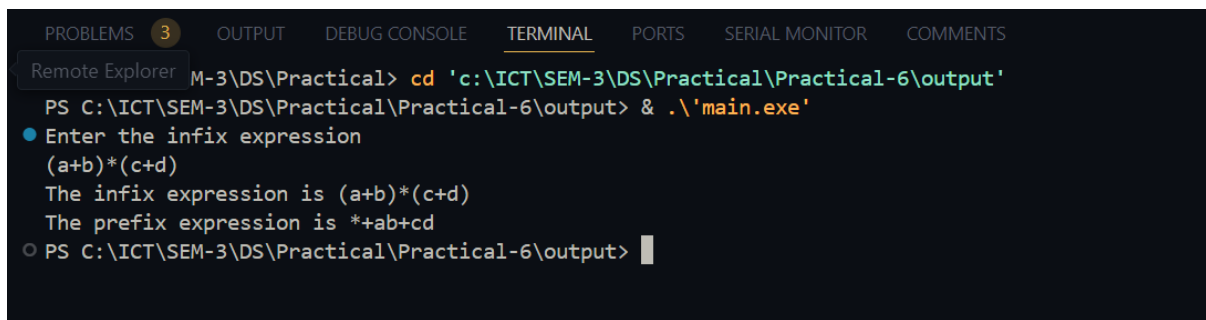
    printf("Enter the infix expression\n");
    gets(infix); // Take infix expression input

    printf("The infix expression is %s\n", infix);

    infixtoprefix(infix, prefix); // Convert infix to prefix

    printf("The prefix expression is %s\n", prefix); // Print the resulting
prefix expression

    return 0;
}
```

Output-

```
PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL PORTS SERIAL MONITOR COMMENTS
Remote Explorer M-3\DS\Practical> cd 'c:\ICT\SEM-3\DS\Practical\Practical-6\output'
PS C:\ICT\SEM-3\DS\Practical\Practical-6\output> & .\'main.exe\'
Enter the infix expression
(a+b)*(c+d)
The infix expression is (a+b)*(c+d)
The prefix expression is *+ab+cd
PS C:\ICT\SEM-3\DS\Practical\Practical-6\output>
```