Name: Amaan Mohib

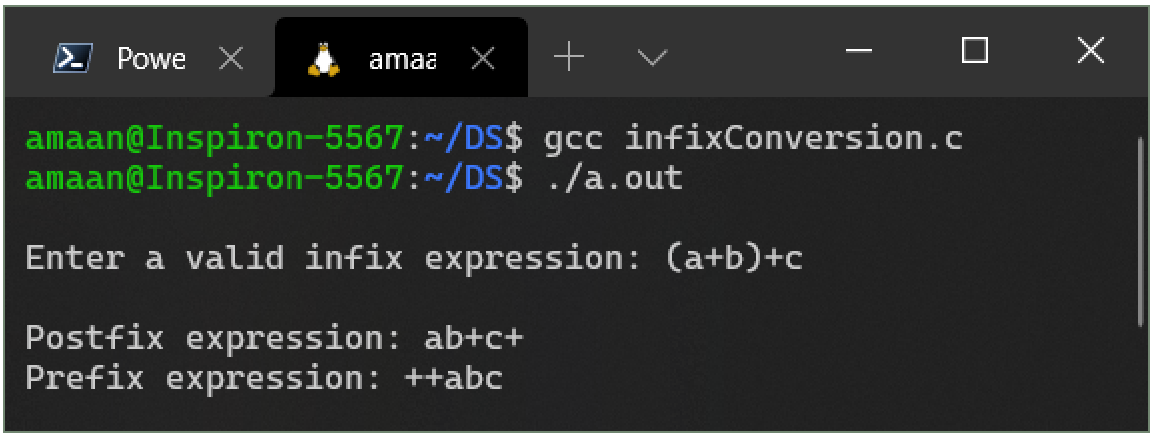
USN: 1NT19IS012

1. Convert infix expression to postfix and prefix expressions

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

#define stack_size 50
int top;
char stack[stack_size], str[20], res[20];

//Pushing into the stack
void push(char ele)
{
    if (top == stack_size - 1)
        printf("\nOverflow\n");
    else
    {
        top++;
        stack[top] = ele;
    }
}

//Popping from the stack
char pop()
{
    if (top == -1)
    {
        printf("\nUnderflow\n");
        return (-1);
    }
    else
    {
        return (stack[top--]);
    }
}

//Postfix conversion
void postfix()
{
    int i = 0, j = 0;
    for (i = 0; str[i] != '\0'; i++) //operand
    {
        if (isalpha(str[i]) || isdigit(str[i]))
            res[j++] = str[i];
        else
        {
            switch (str[i])
            {
            case '(':
                push(str[i]);
                break;
            case '$':
                while (stack[top] == '$')
                    res[j++] = pop();
                push(str[i]);
                break;
            case '/':
            case '*':
                while (stack[top] == '$' || stack[top] == '/' || stack[top] == '*')
                    res[j++] = pop();
                push(str[i]);
                break;
            case '+':
            case '-':
                while (stack[top] == '$' || stack[top] == '/' || stack[top] == '*'
|| stack[top] == '+' || stack[top] == '-')
                    res[j++] = pop();
                push(str[i]);
                break;
            case ')':
                while (stack[top] != '(')
                    res[j++] = pop();
                pop();
                break;
            default:
                printf("\nCannot identify an operator\n");
            }
        }
    }
    while (top > -1)
    {
        res[j++] = pop();
        res[j] = '\0';
    }
}

//Reversing the string
void reverse(char array[20])
{
    int i, j;
    char temp[25];
    for (i = strlen(array) - 1, j = 0; i + 1 != 0; --i, ++j)
    {
        if (array[i] == '(')
            array[i] = ')';
        else if (array[i] == ')')
            array[i] = '(';
        temp[j] = array[i];
    }
    temp[j] = '\0';
```

```
        strcpy(array, temp);
    }

    //Prefix conversion
    void prefix()
    {
        reverse(str);
        postfix(str);
        reverse(res);
    }

    void main()
    {
        top = -1;
        printf("\nEnter a valid infix expression: ");
        scanf("%s", str);
        postfix(str);
        printf("\nPostfix expression: %s", res);
        prefix(str);
        printf("\nPrefix expression: %s\n", res);
    }
```

Output:



```
amaan@Inspiron-5567:~/DS$ gcc infixConversion.c
amaan@Inspiron-5567:~/DS$ ./a.out

Enter a valid infix expression: (a+b)+c

Postfix expression: ab+c+
Prefix expression: ++abc
```