Shreyas Sawant
D7A - 55

AIM: To convert an Infix expression to Postfix expression using Stack ADT

THEORY:

- Stack:

A stack is an ordered collection of items where the addition of new items and removal of existing items take place at same end. This end is commonly referred to as 'TOP' and opposite end as 'BASE'.

The base of stack is significant since items stored in the stack that are closer to base represent those that have been in the stack of the longest. The most recently added item is the one that is in position to be removed first. This is known as LIFO, Last In First Out.

- Rules to convert infix to postfix.

Step 1: Add ')' to end of infix expression

Step 2: Push '(' to stack

Step 3: Repeat until each character in infix is scanned

→ If a '(' is encountered, push it on the stack

→ If an operand is encountered (~~[illegible]~~alphanumeric), add it to postfix expression

→ If a ')' is encountered, then;

a. Repeatedly pop from stack and add to postfix expression until '(' is encountered.

b. Discard the '('.

→ If an operator X is encountered, then;

Repeatedly pop from stack and add that operators to the postfix expression which has same precedence or a higher precedence

Step 4: Repeatedly pop from the stack and add it to the postfix expression until stack is empty.

Step 5: Exit.

Example of Infix to Postfix:

Expression	Current Character	Action	Stack	Postfix
a+(b*c)/(d-e))	a	Push to postfix expression	(	a
	+	Push to stack	(+	a
	(	Push to stack	(+(	a
	b	Pass to postfix expression	(+(	ab
	*	Push to stack	(+(*	ab
	c	Pass to postfix expression	(+(*ε	abc
	)	Pop elements till), and append them to postfix	(+	abc*
	/	Push to stack	(+/	abc*
	(	Push to stack	(+/(	abc*
	d	Pass to postfix expression	(+/(	abc*d
	–	Push to stack	(+/(-	abc*d
	e	Pass to postfix expression	(+/(-	abc*de
	)	Pop elements till) and append them to postfix	(+/	abc*de-
	)	Pop elements till) and append them to postfix		abc*de-/+

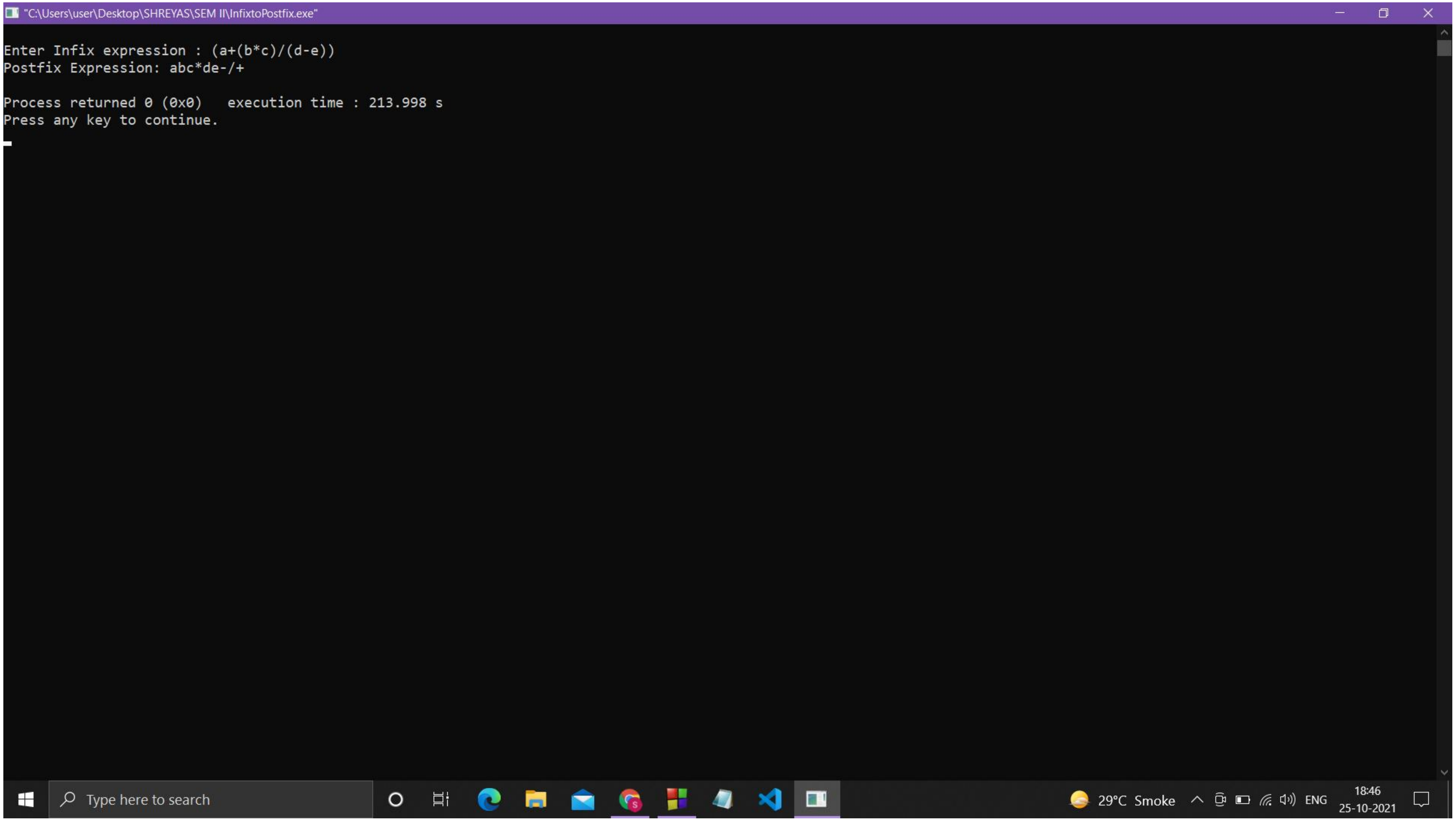
CONCLUSION:

Errors encountered

1) Using incorrect quotes : push ("(');
Solution Correct syntax : push ('(');

2) Incorrect syntax for ~~ext~~ exit : exit 1;
Solution Correct syntax : exit (1);

```c
//SHREYAS SAWANT D7A 55
//Infix to Postfix Conversion

#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>

#define Size 100
int top=-1;
int stack[Size];
void push(int c)
{
    if(top==Size-1)
    {
        printf("OVERFLOW\n");
    }
    else
    {
        top++;stack[top]=c;
    }
}
char pop()
{
    char item ;
    if(top <0)
    {
        printf("stack under flow: invalid infix expression");
        getchar();
        exit(1);
    }
    else
    {
        item = stack[top];
        top = top-1;
        return(item);
    }
}
int is_operator(char symbol)
{   if(symbol == '^' || symbol == '*' || symbol == '/' || symbol == '+' || symbol =='-')
    {
        return 1;
    }
    else
    {
    return 0;
    }
}
int precedence(char symbol)
{
    if(symbol == '^')
    {
        return(3);
    }
    else if(symbol == '*' || symbol == '/')
    {
        return(2);
    }
    else if(symbol == '+' || symbol == '-')
    {
        return(1);
    }
    else
    {
        return(0);
    }
}
void convert(char infix_exp[], char postfix_exp[])
{
    int i, j;
    char item;
    char x;

    push('(');
    strcat(infix_exp,")");
    i=0; j=0;
    item=infix_exp[i];

    while(item != '\0')
    {
        if(item == '(')
        {
            push(item);
        }
        else if( isdigit(item) || isalpha(item))
        {
            postfix_exp[j] = item;
            j++;
        }
        else if(is_operator(item) == 1)
        {
            x=pop();
            while(is_operator(x) == 1 && precedence(x)>= precedence(item))
            {
                postfix_exp[j] = x;
                j++;
                x = pop();
            }
            push(x);
            push(item);                     }
        else if(item == ')')
        {
            x = pop();
            while(x != '(')
            {
                postfix_exp[j] = x;
                j++;
                x = pop();
            }
            }
        else
        {
            printf("\nInvalid infix Expression.\n");
                        getchar();
            exit(1);
        }
        i++;
        item = infix_exp[i];
    }
    if(top>0)
    {
        printf("\nInvalid infix Expression.\n");
        exit(1);
    }
    postfix_exp[j] = '\0';
}

int main()
{
    char infix[Size], postfix[Size];

    printf("\nEnter Infix expression : ");
    gets(infix);

    convert(infix,postfix);
    printf("Postfix Expression: ");
    puts(postfix);
    return 0;
}

```



```
Enter Infix expression : (a+(b*c)/(d-e))
Postfix Expression: abc*de-/+

Process returned 0 (0x0)   execution time : 213.998 s
Press any key to continue.
```