19BCE2484

Sashank Rijal

DSA

1. **Convert the following infix expression to postfix:**

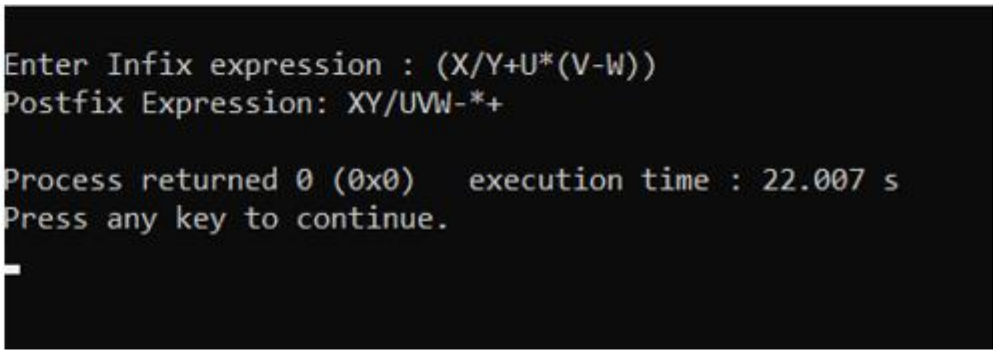
   CODE:

```
#include<stdio.h>

#include<ctype.h>

#include<stdlib.h>

#include<string.h>

#define SIZE 100


char stack[SIZE];

int top = -1;


void push(char item)

{

	if (top >= SIZE - 1)

	{

		printf("\nStack Overflow.");

	}

	else

	{

		top = top + 1;

		stack[top] = item;
	}
}

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 InfixToPostfix(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];

        }
```

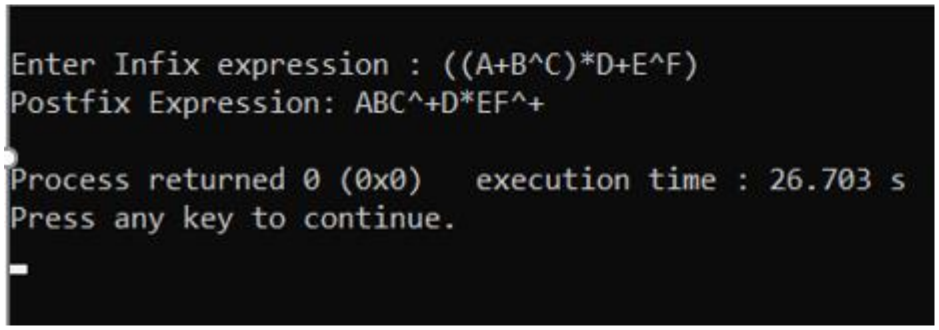
```
	if (top > 0)
	{
		printf("\nInvalid infix Expression.\n");
		getchar();
		exit(1);
	}
	if (top > 0)
	{
		printf("\nInvalid infix Expression.\n");
		getchar();
		exit(1);
	}
	postfix_exp[j] = '\0';
}

int main()
{
	char infix[SIZE], postfix[SIZE];
	printf("\nEnter Infix expression : ");
	gets(infix);
	InfixToPostfix(infix, postfix);
	printf("Postfix Expression: ");
	puts(postfix);
	return 0;
}
```

A. (X/Y + U * (V-W)):



```
Enter Infix expression : (X/Y+U*(V-W))
Postfix Expression: XY/UVW-*+

Process returned 0 (0x0)   execution time : 22.007 s
Press any key to continue.
```

B. ((A + B ^ C) * D + E ^ F):



```
Enter Infix expression : ((A+B^C)*D+E^F)
Postfix Expression: ABC^+D*EF^+

Process returned 0 (0x0)   execution time : 26.703 s
Press any key to continue.
```

2. Evaluate the following postfix expression:

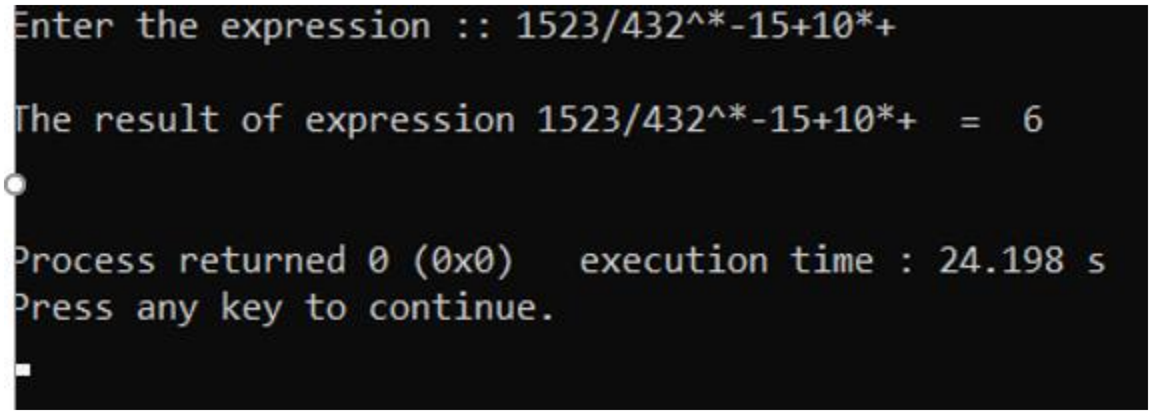
CODE:

```
#include<stdio.h>
int stack[20];
int top = -1;

void push(int x)
{
   stack[++top] = x;
}

int pop()
{
   return stack[top--];
}
```

```
int main()
{
    char exp[20];
    char *e;
    int n1,n2,n3,num;
    printf("Enter the expression :: ");
    scanf("%s",exp);
    e = exp;
    while(*e != '\0')
    {
        if(isdigit(*e))
        {
            num = *e - 48;
            push(num);
        }
        else
        {
            n1 = pop();
            n2 = pop();
            switch(*e)
            {
            case '+':
            {
                n3 = n1 + n2;
                break;
            }
            case '-':
            {
                n3 = n2 - n1;
                break;
            }
            case '*':
            {
                n3 = n1 * n2;
                break;
            }
            case '/':
            {
                n3 = n2 / n1;
                break;
            }
            }
            push(n3);
        }
        e++;
    }
    printf("\nThe result of expression %s  =  %d\n\n",exp,pop());
    return 0;
}
```

Expression : 15 2 3 / 4 3 2 ^ * - 15 + 10 * +

OUTPUT:



```
Enter the expression :: 1523/432^*-15+10*+

The result of expression 1523/432^*-15+10*+  =  6


Process returned 0 (0x0)   execution time : 24.198 s
Press any key to continue.
```