Name : Hemal Nisar

Div: C2, Batch : C24

Roll Number : 2003126

DS LAB EXPERIMENT NO. 3

Aim : To evaluate Postfix Expression using stack ADT.

Theory: Postfix Evaluation Pseudocode:

Input is taken in the form of Infix or postfix.
Infix input is converted to postfix.

Loop(till end of postfix) {

If character is a digit :
Push(character)
Else{
Operand 2 =pop(); //Top 2 operands are popped.
Operand 1= pop)();
int value;

switch(character which is an operator)
case1 '+'

value =operand 1 +operand 2;
break;
same for other operands;
}//end of switch case
Push(value)
}//end of while loop
Printf(“final results :%d”,pop());
// The final element of the stack is the answer after the loop.

Example:

Infix expression: (8*3)/2
Postfix expression : 83*2/

Token	Stack
8	8
3	8,3
*	24
2	24 , 2
/	12

PROGRAM :

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.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;
}
}
void pushy(int y)
{
stack[++Top] = y;
}
void print_stack()
{
int i;
for( i = 0; i <= Top; i++){
printf("%d,", stack[i]);
}
}
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 popy()
{
return stack[Top--];
}
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, k;
```

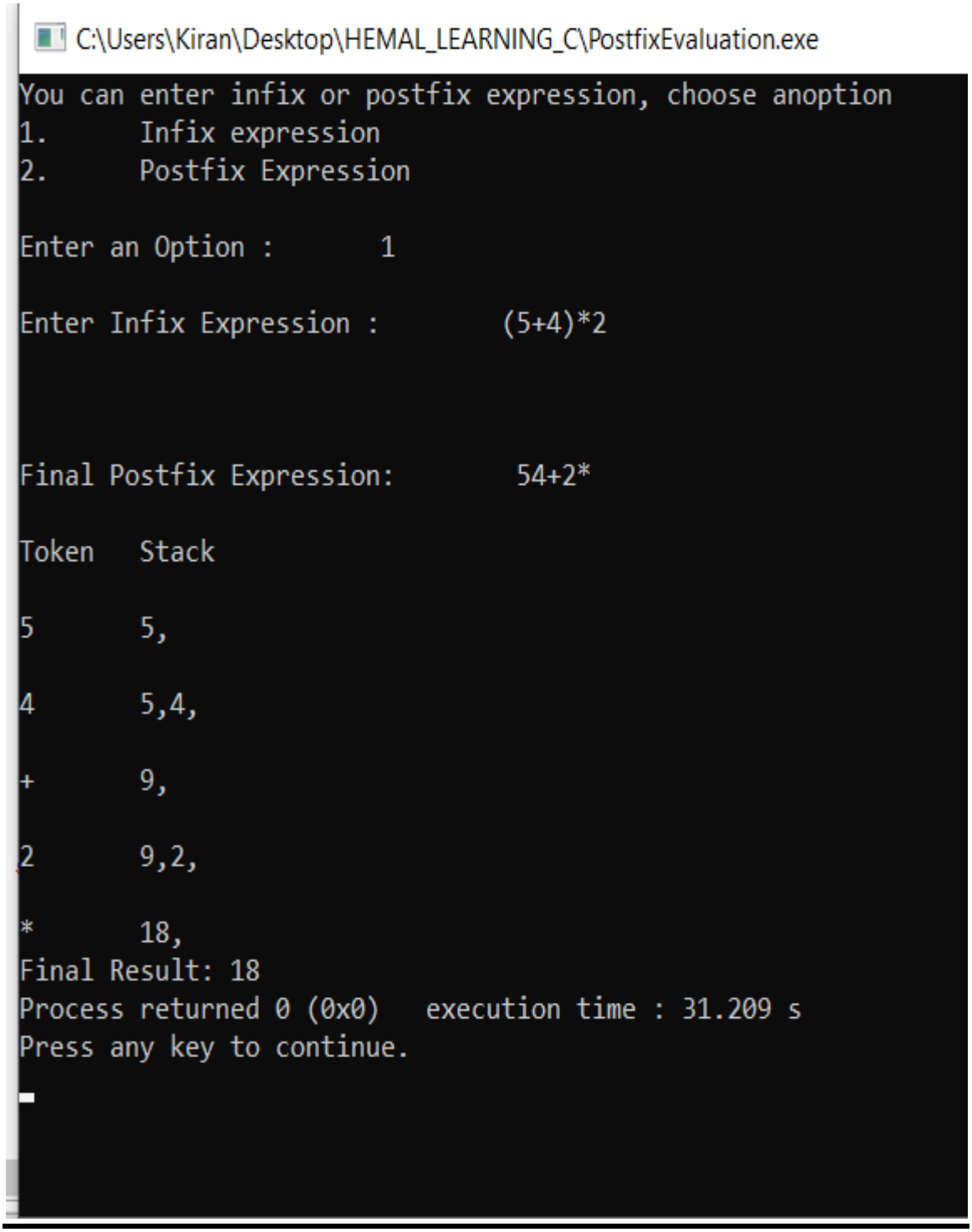
```
char Item;
char x;
push('(');
strcat(infix_exp, ")");
i = 0;
k = 0;
Item = infix_exp[i];
while (Item != '\0')
{
if (Item == '('){
push(Item);
}
else if (isalnum(Item)){
postfix_exp[k] = Item;
k++;
}
else if (is_operator(Item) == 1)
{
x = pop();
while (is_operator(x) == 1 && precedence(x) >=
precedence(Item))
{
postfix_exp[k] = x;
k++;
x = pop();
}
push(x);
push(Item);
}
else if (Item == ')'){
x = pop();
while (x != '('){
postfix_exp[k] = x;
k++;
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);
}
postfix_exp[k] = '\0';
}
int main()
{
int i, h;
char exp[SIZE];
char *e;
int n1,n2,n3,num;
char infix[SIZE], postfix[SIZE];
printf("You can enter infix or postfix expression, choose an
option\n1.\tInfix expression\n2.\tPostfix Expression\n\nEnter
an
Option :\t");
scanf("%d", &h);
switch(h){
case 1:
for (i = 0; i < SIZE; i++){
postfix[i] = '\0';
}
printf("\nEnter Infix Expression :\t");
```

```
scanf("%s",&infix);
printf("\n\n");
InfixToPostfix(infix, postfix);
printf("\n");
printf("Final Postfix Expression:\t ");
puts(postfix);
e = postfix;
break;
case 2:
printf("Enter Postfix Expression :\t");
scanf("%s",&postfix);
e = postfix;
break;
}
printf("\nToken\tStack\n");
char token;
while(*e != '\0')
{
if(isdigit(*e))
{
num = *e - '0';
token = *e;
pushy(num);
}
else
{
n1 = popy();
n2 = popy();
switch(*e)
{
case '+':
{
n3 = n1 + n2;
token = '+';
break;
```

```
}
case '-':
{
n3 = n2 - n1;
token = '-';
break;
}
case '*':
{
n3 = n1 * n2;
token = '*';
break;
}
case '/':
{
n3 = n2 / n1;
token='/';
break;
}
}
pushy(n3);
}
printf("\n%c\t",token);
print_stack();
printf("\n");
e++;
}
printf("Final Result: %d",popy());
return 0;
}
```

OUTPUT:


C:\Users\Kiran\Desktop\HEMAL_LEARNING_C\PostfixEvaluation.exe
You can enter infix or postfix expression, choose anoption
1. Infix expression
2. Postfix Expression
Enter an Option : 1
Enter Infix Expression : (5+4)*2
Final Postfix Expression: 54+2*
Token Stack
5 5,
4 5,4,
+ 9,
2 9,2,
* 18,
Final Result: 18
Process returned 0 (0x0) execution time : 31.209 s
Press any key to continue.