**Data Structures in C LAB – {1}**

Akshaya agarwal

2247207

3-MCA B

**CODE :**

#include<stdio.h>

#include<stdlib.h>

#include<ctype.h>

#include<string.h>

#define SIZE 50

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 Underflow");

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[], char postfix[])

{

int i, j;

char item;

char x;

push('(');

strcat(infix,")");

i=0;

j=0;

item=infix[i];

while(item != '\0')

{

if(item == '(')

push(item);

else if( isdigit(item) || isalpha(item))

{

postfix[j] = item;

j++;

}

else if(is\_operator(item) == 1)

{

x = pop();

while(is\_operator(x) == 1 && precedence(x)>= precedence(item))

{

postfix[j] = x;

j++;

x = pop();

}

push(x);

push(item);

}

else if(item == ')')

{

x = pop();

while(x != '(')

{

postfix[j] = x;

j++;

x = pop();

}

}

else

{

printf("\nInvalid infix Expression.\n");

getchar();

exit(1);

}

i++;

item = infix[i];

}

if(top>0)

{

printf("\nInvalid infix Expression.\n");

getchar();

exit(1);

}

postfix[j] = '\0';

}

int main()

{

char infix[SIZE], postfix[SIZE];

int ch, n=0;

while(n==0)

{

printf("\n1.Enter Infix expression\n2.Print Postfix Expression: \n3.Exit");

printf("\nenter your choice\n");

scanf("%d", &ch);

switch(ch)

{

case 1:

fflush(stdin);

printf("\nEnter Infix expression : ");

gets(infix);

InfixToPostfix(infix,postfix);

break;

case 2:

printf("Postfix Expression: ");

puts(postfix);

break;

case 3:

n=1;

break;

default:

printf("invalid choice");

}

}

/\*printf("\nEnter Infix expression : ");

gets(infix);

InfixToPostfix(infix,postfix);

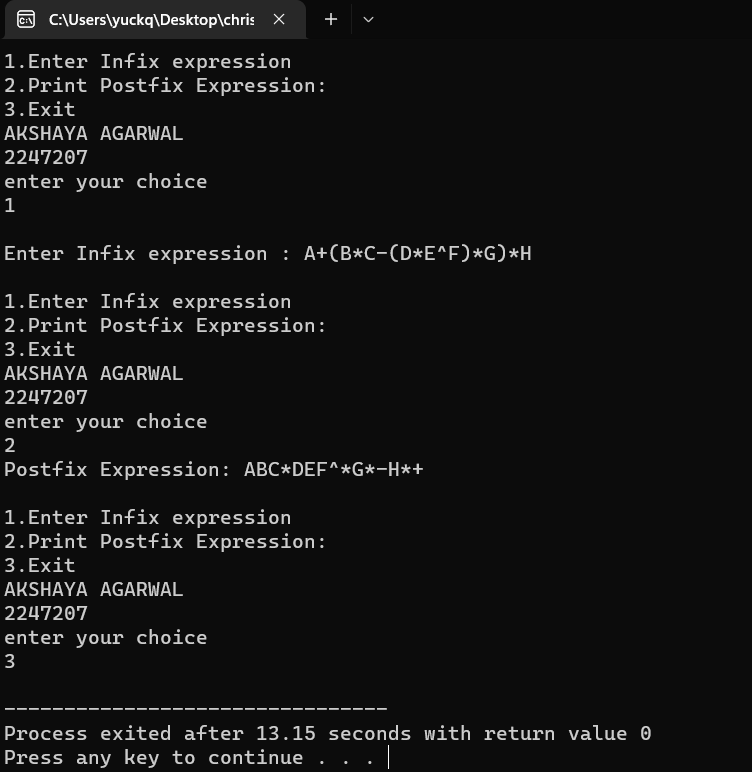
printf("Postfix Expression: ");

puts(postfix);\*/

return 0;

}

**OUTPUT :**

****