**Experiment No. 2b**

**Title :** Conversion of infix to postfix expression

**Problem Statement :** Convert an infix expression into a postfix expression

**Algorithm :**

**S1 :** Start

**S2 :** Declare an array along with the size, a top variable to keep a track of the index of stack.

**S3 :** Declare a while loop and if the character is a number or an alphabet the print it

**S4 :** If the character is ‘(‘ then push it in stack

**S5 :** If the character is ‘)’ the pop until the character is ‘(‘ and print it

**S6 :** Declare priority function and ‘(‘ having least priority and ‘+’ and ‘-‘ medium priority and ‘\*’ and ‘/’ with highest priority.

**S7 :** If the top value in stack has highest priority then pop else push the character.

**S8 :** Continue this till the end of the string and print the output at last using while loop

**S9 :** Stop

**Code :**

#include<stdio.h>

char stack[20];

int top = -1;

void push(char x)

{

stack[++top] = x;

}

char pop()

{

if(top == -1)

return -1;

else

return stack[top--];

}

int priority(char x)

{

if(x == '(')

return 0;

if(x == '+' || x == '-')

return 1;

if(x == '\*' || x == '/')

return 2;

}

main()

{

char exp[20];

char \*e, x;

printf("Enter the expression :: ");

scanf("%s",exp);

e = exp;

while(\*e != '\0')

{

if(isalnum(\*e))

printf("%c",\*e);

else if(\*e == '(')

push(\*e);

else if(\*e == ')')

{

while((x = pop()) != '(')

printf("%c", x);

}

else

{

while(priority(stack[top]) >= priority(\*e))

printf("%c",pop());

push(\*e);

}

e++;

}

while(top != -1)

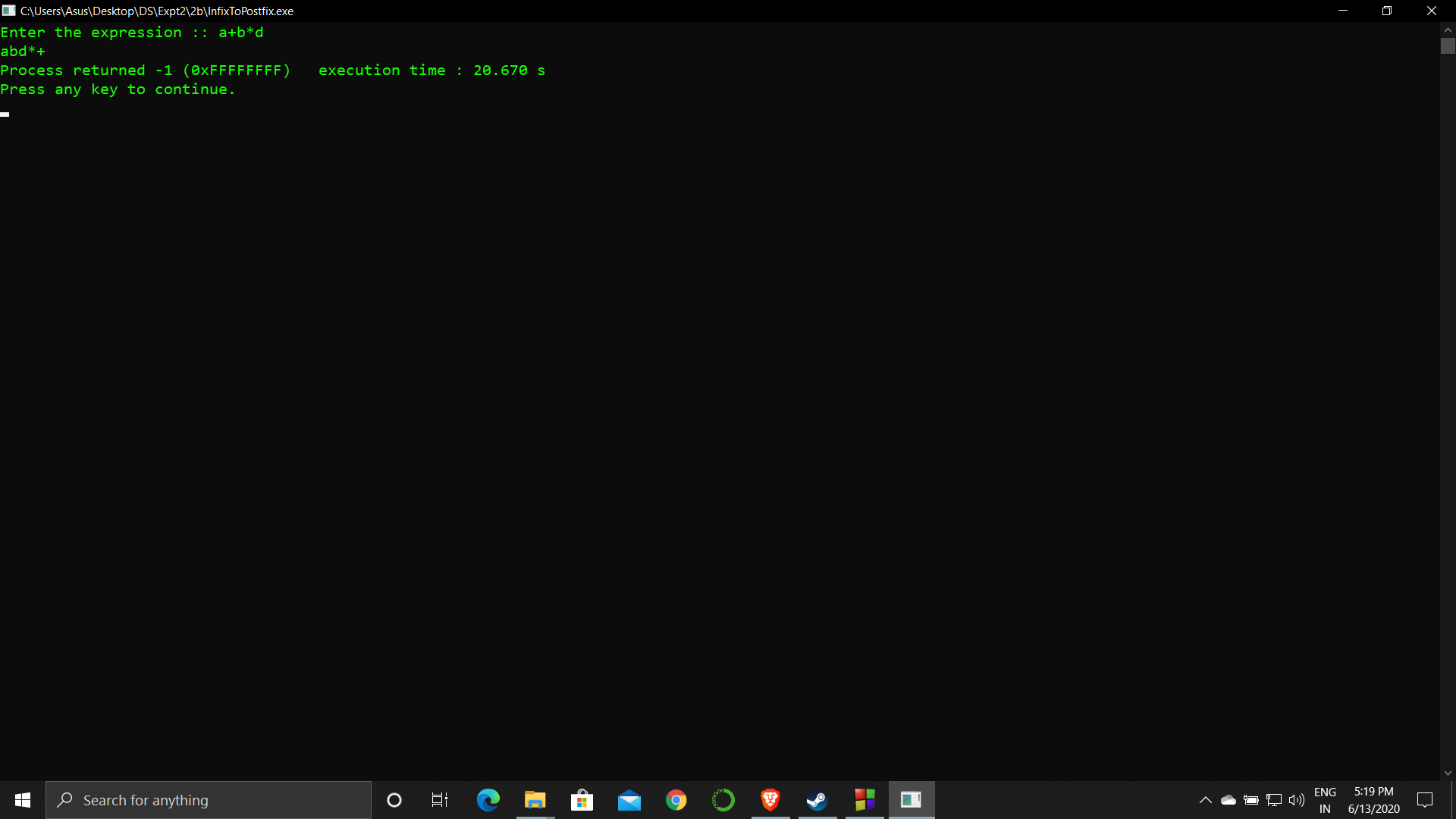
{

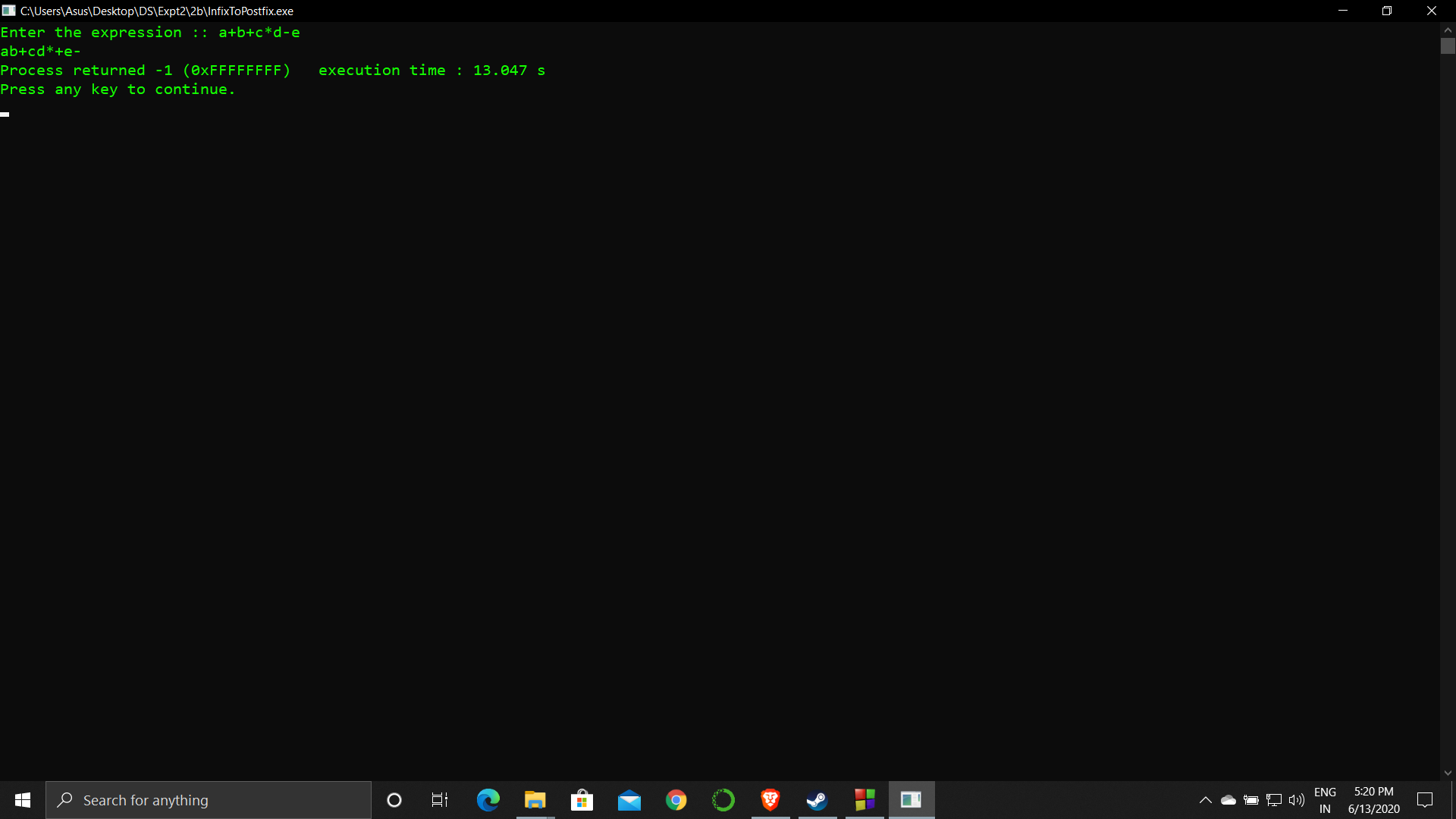
printf("%c",pop());

}

}

**Output :**

****

****

**Analysis :**

The program only generates a postfix expression but does not evaluate it.

The program is restricted to operators like ‘+’,’-‘,’\*’,’/’ only.

Stack size I restricted which restricts its input and output string size.