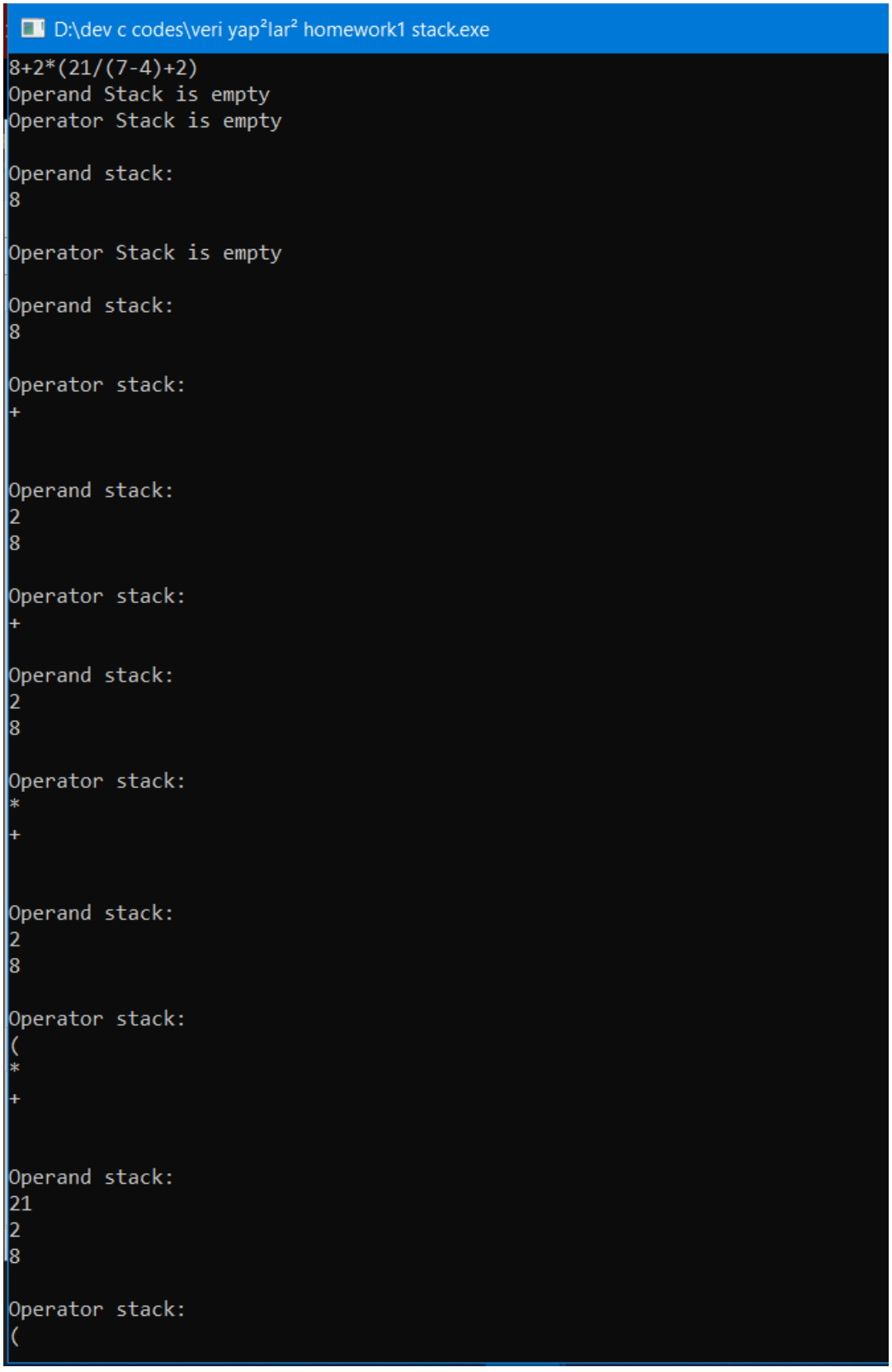
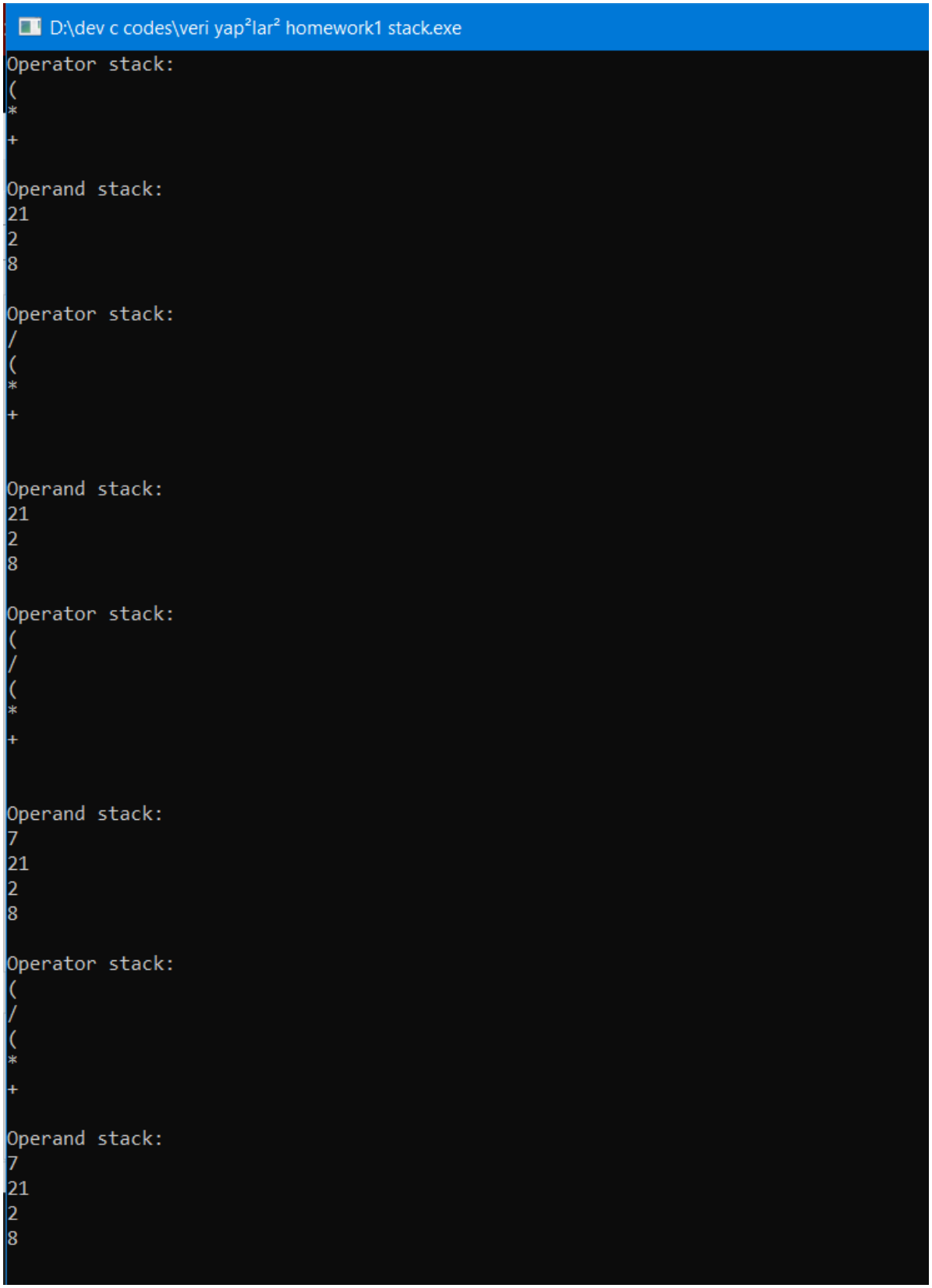
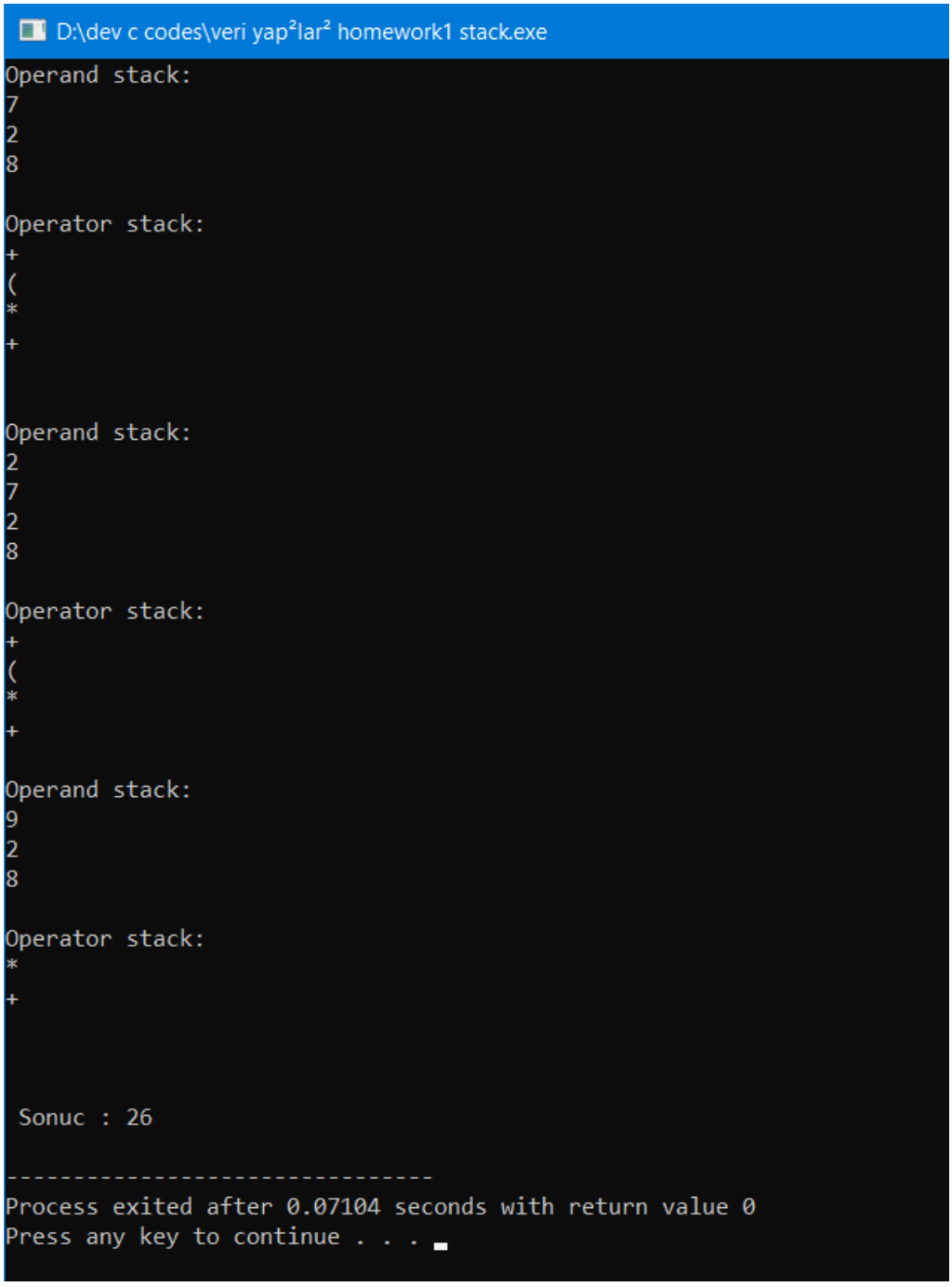
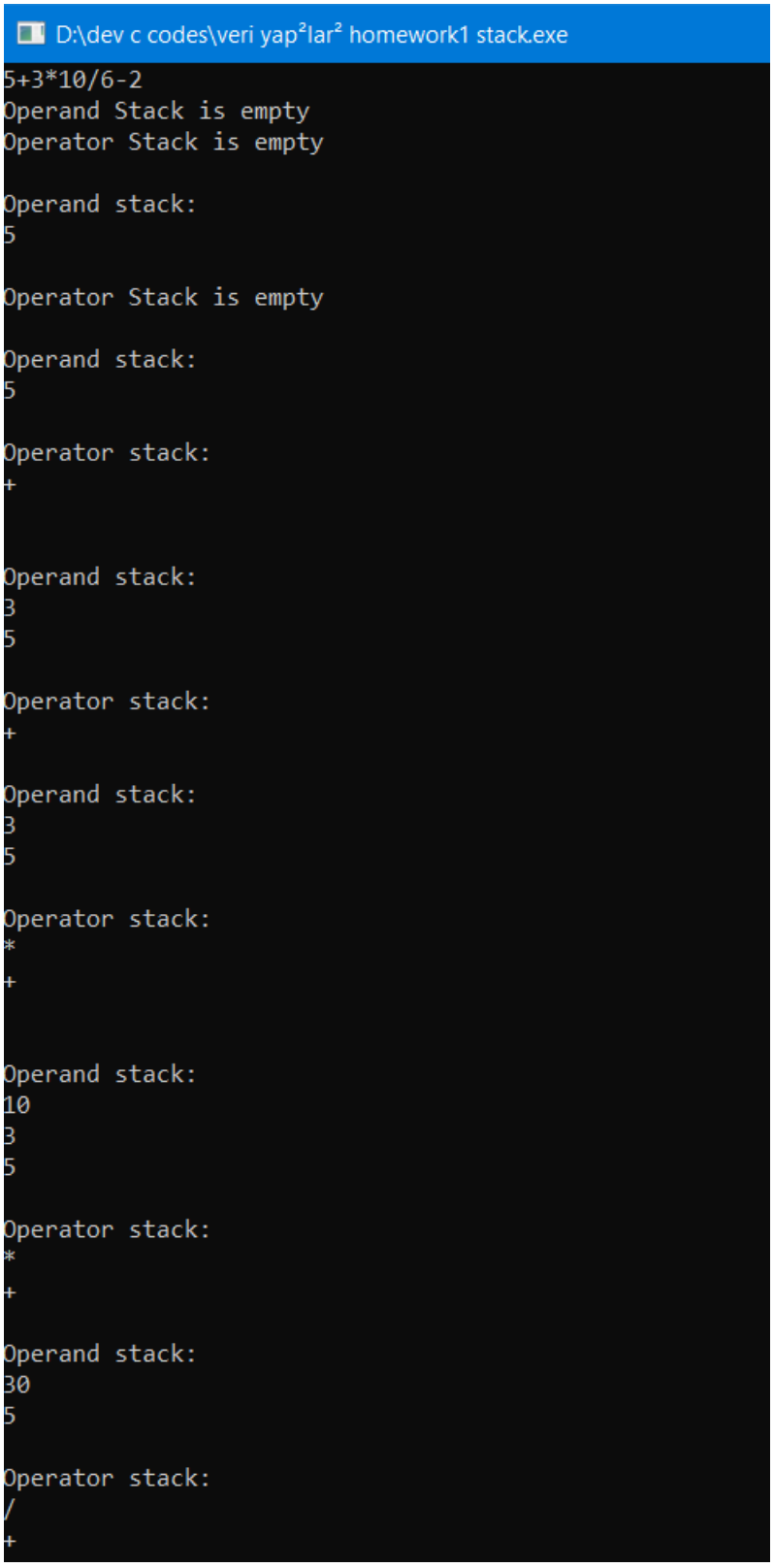
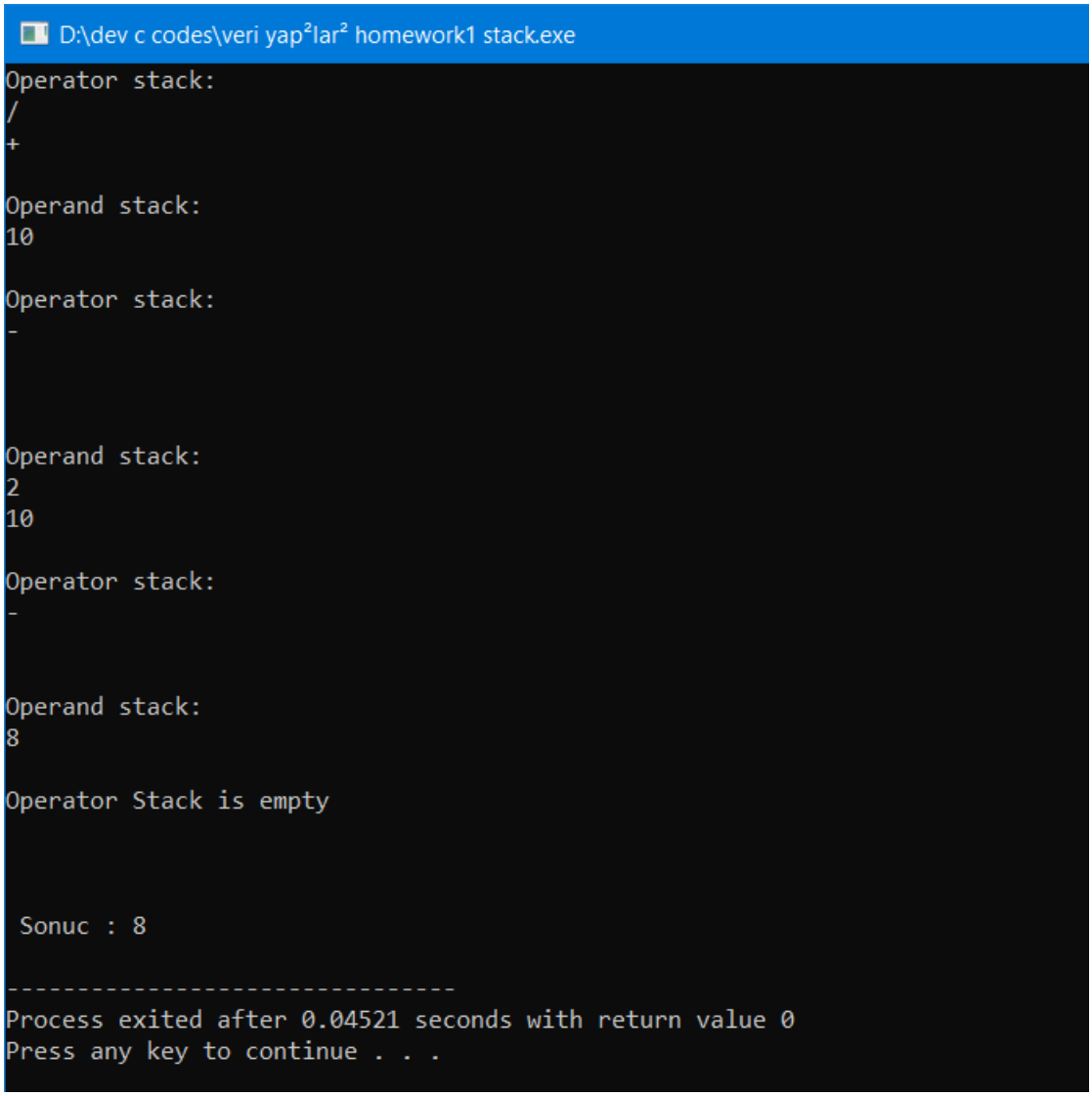

D:\dev c codes\veri yap²lar² homework1 stack.exe
8+2*(21/(7-4)+2)
Operand Stack is empty
Operator Stack is empty
Operand stack:
8
Operator Stack is empty
Operand stack:
8
Operator stack:
+
Operand stack:
2
8
Operator stack:
+
Operand stack:
2
8
Operator stack:
*
+
Operand stack:
2
8
Operator stack:
(
*
+
Operand stack:
21
2
8
Operator stack:
(


D:\dev c codes\veri yap²lar² homework1 stack.exe
Operator stack:
(
*
+
Operand stack:
21
2
8
Operator stack:
/
(
*
+
Operand stack:
21
2
8
Operator stack:
(
/
(
*
+
Operand stack:
7
21
2
8
Operator stack:
(
/
(
*
+
Operand stack:
7
21
2
8

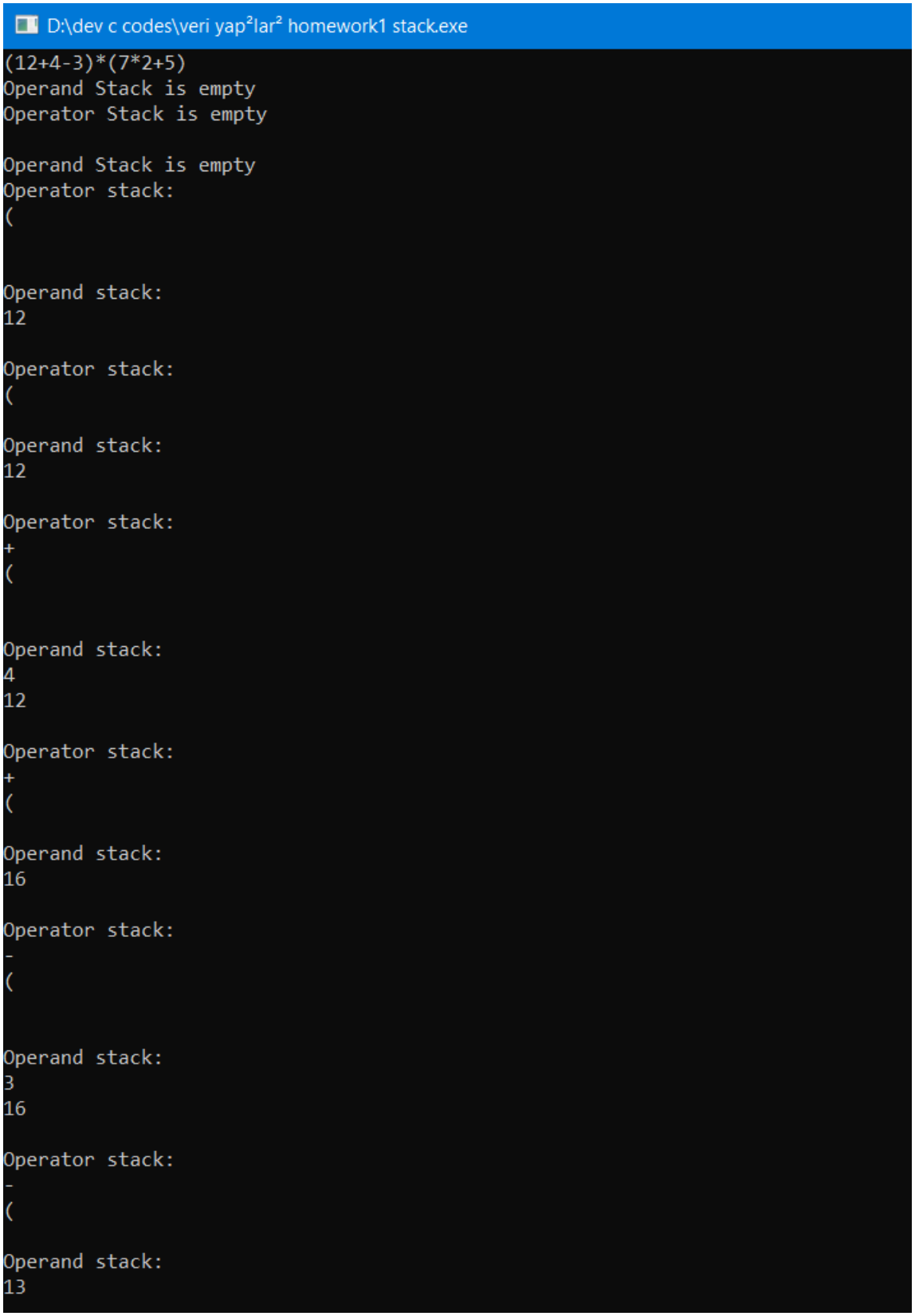

D:\dev c codes\veri yap²lar² homework1 stack.exe
Operand stack:
7
2
8
Operator stack:
+
(
*
+
Operand stack:
2
7
2
8
Operator stack:
+
(
*
+
Operand stack:
9
2
8
Operator stack:
*
+
Sonuc : 26

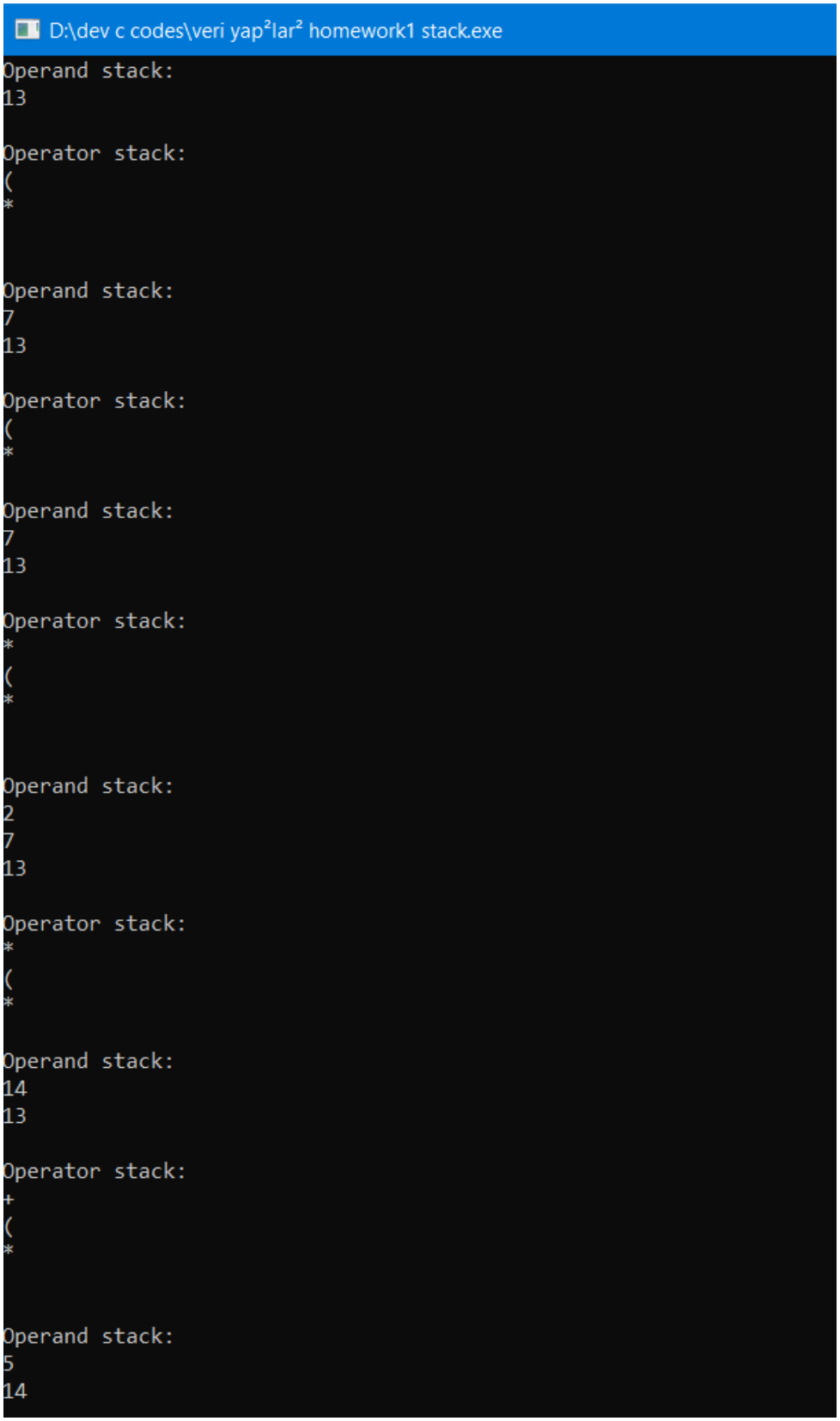
Process exited after 0.07104 seconds with return value 0
Press any key to continue . . .

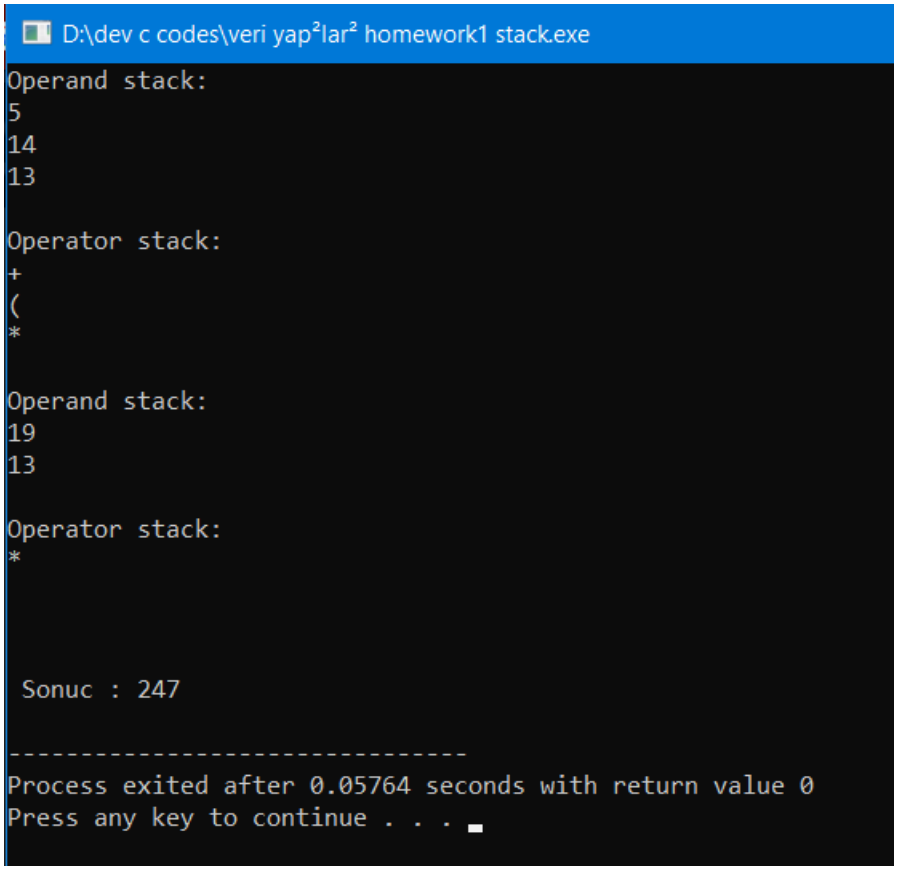

D:\dev c codes\veri yap²lar² homework1 stack.exe
5+3*10/6-2
Operand Stack is empty
Operator Stack is empty
Operand stack:
5
Operator Stack is empty
Operand stack:
5
Operator stack:
+
Operand stack:
3
5
Operator stack:
+
Operand stack:
3
5
Operator stack:
*
+
Operand stack:
10
3
5
Operator stack:
*
+
Operand stack:
30
5
Operator stack:
/
+


D:\dev c codes\veri yap²lar² homework1 stack.exe
Operator stack:
/
+
Operand stack:
10
Operator stack:
-
Operand stack:
2
10
Operator stack:
-
Operand stack:
8
Operator Stack is empty
Sonuc : 8

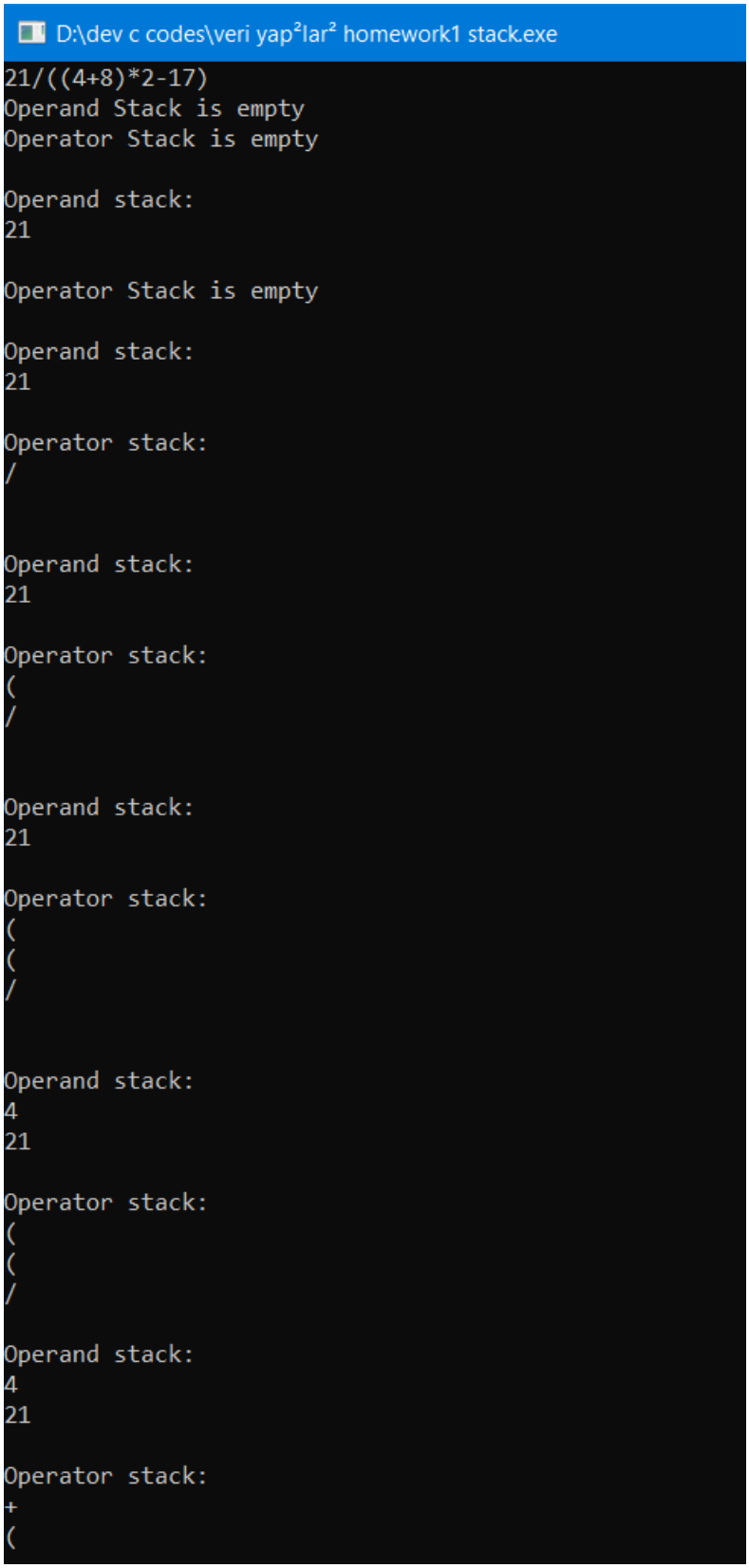
Process exited after 0.04521 seconds with return value 0
Press any key to continue . . .

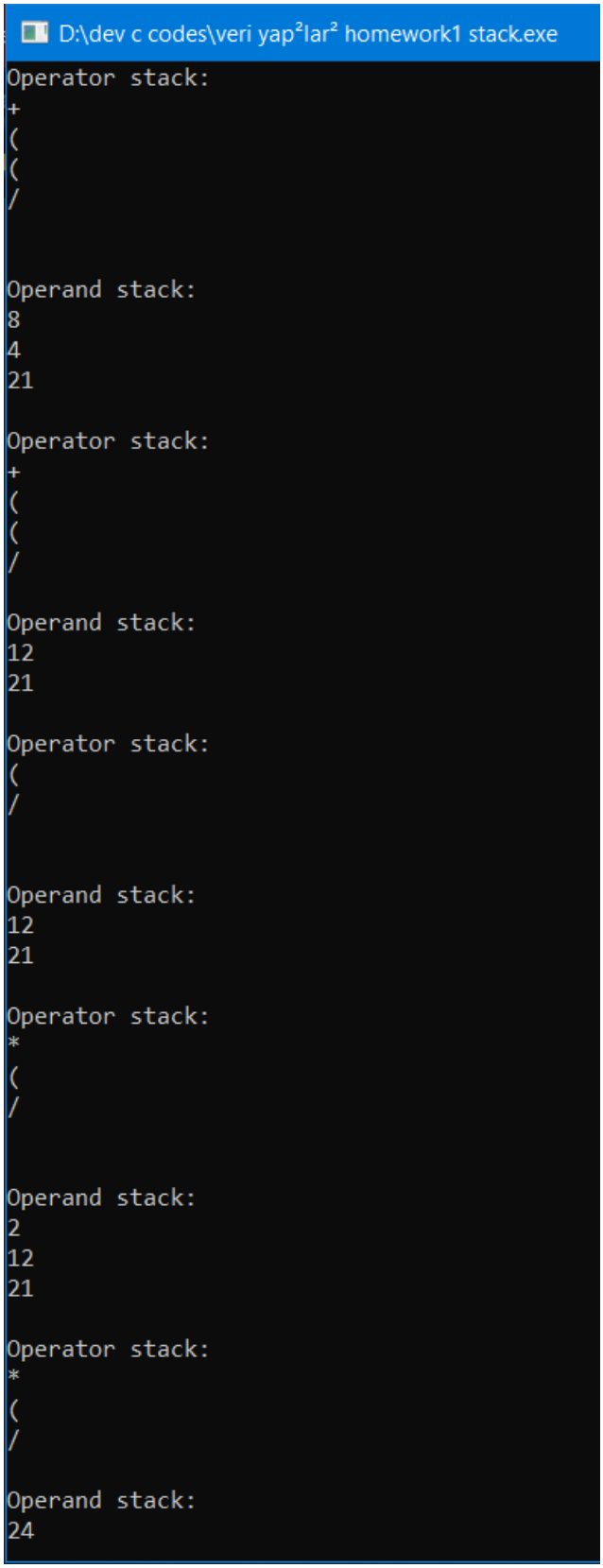

D:\dev c codes\veri yap²lar² homework1 stack.exe
(12+4-3)*(7*2+5)
Operand Stack is empty
Operator Stack is empty
Operand Stack is empty
Operator stack:
(
Operand stack:
12
Operator stack:
(
Operand stack:
12
Operator stack:
+
(
Operand stack:
4
12
Operator stack:
+
(
Operand stack:
16
Operator stack:
-
(
Operand stack:
3
16
Operator stack:
-
(
Operand stack:
13

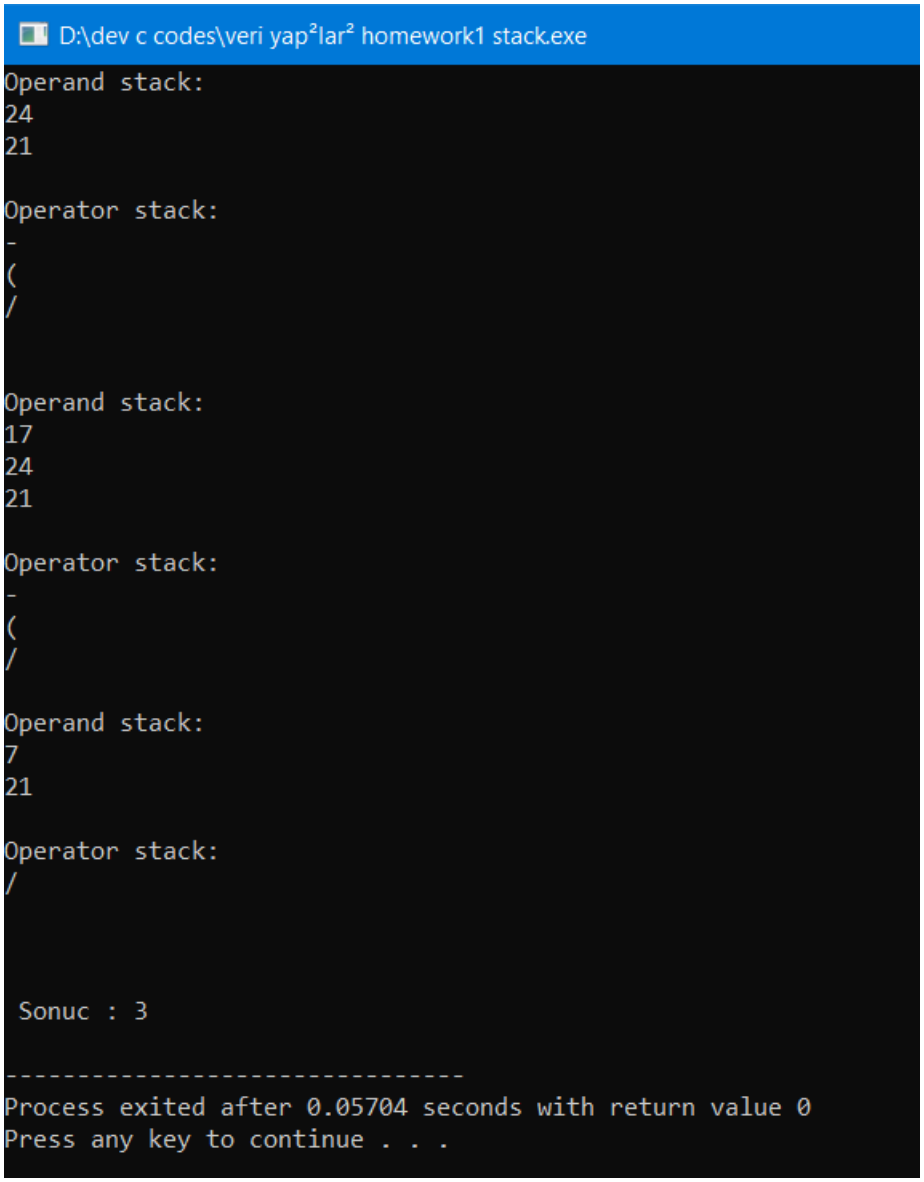

D:\dev c codes\veri yap²lar² homework1 stack.exe
Operand stack:
13
Operator stack:
(
*
Operand stack:
7
13
Operator stack:
(
*
Operand stack:
7
13
Operator stack:
*
(
*
Operand stack:
2
7
13
Operator stack:
*
(
*
Operand stack:
14
13
Operator stack:
+
(
*
Operand stack:
5
14


D:\dev c codes\veri yap²lar² homework1 stack.exe
Operand stack:
5
14
13
Operator stack:
+
(
*
Operand stack:
19
13
Operator stack:
*
Sonuc : 247

Process exited after 0.05764 seconds with return value 0
Press any key to continue . . .


D:\dev c codes\veri yap²lar² homework1 stack.exe
21/((4+8)*2-17)
Operand Stack is empty
Operator Stack is empty
Operand stack:
21
Operator Stack is empty
Operand stack:
21
Operator stack:
/
Operand stack:
21
Operator stack:
(
/
Operand stack:
21
Operator stack:
(
(
/
Operand stack:
4
21
Operator stack:
(
(
/
Operand stack:
4
21
Operator stack:
+
(


D:\dev c codes\veri yap²lar² homework1 stack.exe
Operator stack:
+
(
(
/
Operand stack:
8
4
21
Operator stack:
+
(
(
/
Operand stack:
12
21
Operator stack:
(
/
Operand stack:
12
21
Operator stack:
*
(
/
Operand stack:
2
12
21
Operator stack:
*
(
/
Operand stack:
24



Yukarıda ekran çıktıları verilen işlemler sırası ile;

8 + 2 * (21 / (7 – 4) + 2)

5 + 3 * 10 / 6 – 2

(12 + 4 - 3) * (7 * 2 + 5)

21 / ((4 + 8) * 2 – 17)

Struct yapısı kullanılarak operand ve operatörleri tutması için 2 ayrı stack oluşturulmuştur.

Main fonksiyonu içinde kullanıcıdan alınan string, sayılar ve işaretler olarak ayrılıyor. Sayıların tek basamaklı olma zorunluluğu yok, birden çok basamaklı olması durumunda count değişkeniyle tuttuğumuz basamak sayısına göre sayı hesaplanıp gerçek değeriyle sayı yığınına atılıyor.

Stringde okunan değer işaret ise check() fonksiyonuna gidiyor. Burada gelen işaret ve stack durumuna bakılarak, operatöre göre işlem yapan calculate() ve işlem önceliğini koruyan prec() fonksiyonları yardımıyla gerekli matematiksel işlemler ve stack için push-pop işlemleri yapılıyor.

String okuma işlemi bittikten sonra operatör stacki boş değilse içindeki işaretler calculate() fonksiyonuna gönderilip gerekli işlemler tamamlanıyor.

Nihai sonuç operand stackinden pop edilip ekrana yazdırılıyor.

Program kodu

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include<conio.h>

void check(char);
int prec(char);
void calculate(char);

struct node
        {
           int info;
           struct node *ptr;
        }*top,*top1,*temp;

        int cnt=0;
        int cnt1=0;
        void create(){top=NULL;}

        void push (int data){
                if (top==NULL){
```

```
                top=(struct node *) malloc (sizeof(struct node));
                top->ptr = NULL;
                top->info = data;
        }
        else
        {
                temp=(struct node *) malloc (sizeof(struct node));
                temp->ptr= top;
                temp->info= data;
                top = temp;
        }
        cnt++;
}

int pop(){
        int val;
        top1=top;

        if (top1 == NULL){
                //      printf("Stack is empty\n");
                return 0;
        }
        else
        {
                top1=top1->ptr;
                val=top->info;
                free(top);
                top=top1;
                cnt--;
```

```
                return val;
        }
}

void display(){
        top1=top;
        if (top1==NULL){
                printf("Operand Stack is empty\n");
                return;
        }
        else
        {
                printf("Operand stack:\n");
                while(top1!=NULL){
                        printf("%d \n",top1->info);
                        top1=top1->ptr;
                }
                printf("\n");
        }
}

int isEmpty(){
        if (top==NULL)
                return 0;
        else return 1;
}

int peek(){
        if(isEmpty() == 0)
```

```
                return 0;
        else return top->info;
}



void destroy(){
        top1=top;
        while(top1!=NULL){
                top1=top->ptr;
                free(top);
                top=top1;
                top1=top1->ptr;
        }
free(top1);
top=NULL;
cnt=0;
}

struct node1
{
   char info;
   struct node1 *ptr;
}*topp,*topp1,*tempp;

void create1(){topp=NULL;}

void push1 (char data){
        if (topp==NULL){
```

```
                topp=(struct node1 *) malloc (sizeof(struct node1));
                topp->ptr = NULL;
                topp->info = data;
        }
        else
        {
                tempp=(struct node1 *) malloc (sizeof(struct node1));
                tempp->ptr= topp;
                tempp->info= data;
                topp = tempp;
        }
        cnt1++;
}

char pop1(){
        char val;
        topp1=topp;

        if (topp1 == NULL){
        //      printf("Stack is empty\n");
        }
        else
        {
                topp1=topp1->ptr;
                val=topp->info;
                free(topp);
                topp=topp1;
                cnt1--;
                return val;
```

```
        }
}

void display1(){
        topp1=topp;
        if (topp1==NULL || topp1->info == '\0'){
                printf("Operator Stack is empty\n\n");
                return;
        }
        else
        {       printf("Operator stack:\n");
                while(topp1!=NULL){
                        printf("%c \n",topp1->info);
                        topp1=topp1->ptr;
                }
                printf("\n");
        }
}

int isEmpty1(){
        if (topp==NULL)
        return 0;
        else return 1;
}

char peek1(){
        if(isEmpty1() == 0)
                return 0;
        else return topp->info;
```

```
        }



        void destroy1(){
                topp1=topp;
                while(topp1!=NULL){
                        topp1=topp->ptr;
                        free(topp);
                        topp=topp1;
                        topp1=topp1->ptr;
                }
        free(topp1);
        topp=NULL;
        cnt1=0;
        }


int main()
{
   int sayi[10];
   int i,j,l,k,number,num,count,okk;
   char isaret[20],string[40];
   char op;
   //getch();
   printf("Islemi girin:\n");
   scanf("%s",string);
  printf("%s\n",string);
  display();
```

```
display1();
 i = l = k= 0;
 //printf("%c\n",string[i]);
 while(string[i] != '\0'){
 //     if (string[i] != ' '){

    count = -1; // 10 üssü kuvveti olması için -1 den başlattık
    okk = 0;
    while(string[i] -'0' <= 9 && string[i] -'0'>= 0){ //string[i] sayı ise
       //printf("%c\n",string[i]);
       i++;
       count++; //kaç basamak sayı bulmak için
       okk = 1; //okk 1 ise sayı bulunmuştur.
    }
    j = i - count - 1;
    sayi[l] = 0; //toplamı hesaplamak için tutulan dizi
    while(count != -1){
       num = (int)string[j] - 48;
       sayi[l] = sayi[l] + num * pow(10,count);
       j++;
       count--;
    }
    //printf("%d",number);
  //  printf("ok:%d\n",okk);
                  if(okk == 1){
                              push(sayi[l]);
                              display();
                              display1();
                              l++;
```

```
   } // if string[i] != NULL olması lazım
                isaret[k] = string[i];
   check(isaret[k]);
   display();
                display1();
                k++;
   //printf("%c",op);
   i++;
   printf("\n");
  }

while(isEmpty() != 0 && isEmpty1() != 0){ //2 yığında boş değilse
        op = pop1();
        calculate(op);
        }
        if(isEmpty1() == 0) //gereksiz check pointler
                number = pop();
                if(isEmpty() == 0)
                        printf("\n Sonuc : %d \n",number);
/*for(i=0;i<l;i++)
   printf("%d ",sayi[i]);
printf("\n");
i = 0;
for(i=0;i<k;i++)
   printf("%c ",isaret[i]);
        printf("\n");
//display();
        //display1();*/
```

```
	return 0;
}

int prec(char op){
	if(op == '+' || op == '-')
		return 1;
	if(op == '*' || op == '/')
		return 2;
	if(op == '(' || op == ')')
		return 3;
	return 0;
}

void calculate(char op){ // if a == NULL push b yapması lazım
	int a,b;
	b = pop();
	a = pop();
	if (a == 0){

	push(b);
//	printf("\n a :%d b :%d \n",a,b);
	}
	else{

	if (op == '+'){
		a=a+b;
		push(a);
	}
	if (op == '-'){
```

```
                a=a-b;
                push(a);
        }
        if (op == '*'){
                a=a*b;
                push(a);
        }
        if (op == '/'){
                a=a/b;
                push(a);
        }
        }
}

void check(char op){
        char temp;
        int a,b;
        int i,j;
        if (op == '(')
                push1(op);
        else if(op == ')'){
                temp=pop1();
                while (temp != '(' && temp != '\0'){
                        calculate(temp);
                        temp=pop1();
                }
        }
        else{
                temp=peek1();
```

```
            if(temp == '(')
                push1(op);
            else if (prec(temp) < prec(op)){
                push1(op);
            }
            else if(prec(temp) >= prec(op)){
                temp = pop1();
                i = prec(temp);
                j = prec(op);
                while(i >= j && temp != '(' && temp != '\0'){
                    calculate(temp);
                    temp = pop1();
                    i = prec(temp);
                    j = prec(op);
                }
                push1(temp);
                push1(op);

            }
        }


}
```