```
4572+-*
```

ample Output 0

```
-16
```

BODMAS

+ , −

/ m

$4 * 5 - (7 + 2)$

$4 * (5 - 9)$

$4 * (-4) \Rightarrow -16$

'4' − 48

52 − 48

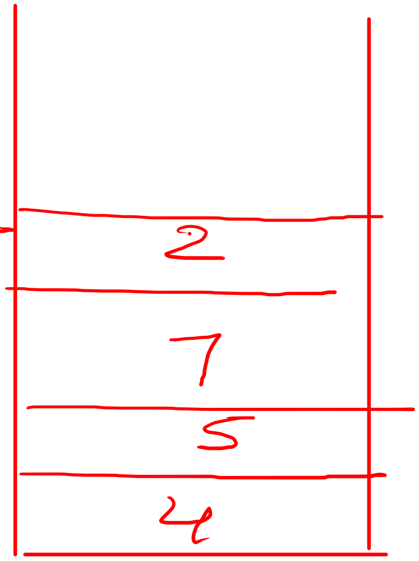
~~$\boxed{4}$~~

53 − 48

$\boxed{5}$

55 − 48

$\boxed{7}$



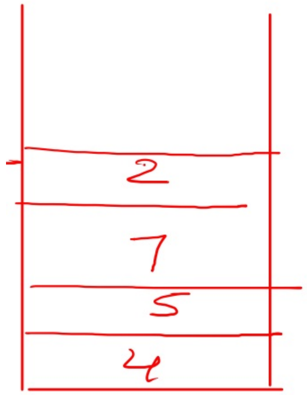
52 − 48 = $\boxed{4}$

string

4572 + − *

Character ('s Digit)

'0'	= 48
'1'	= 49
'2'	= 50
'3'	= 51
'4'	= 52
'5'	= 53



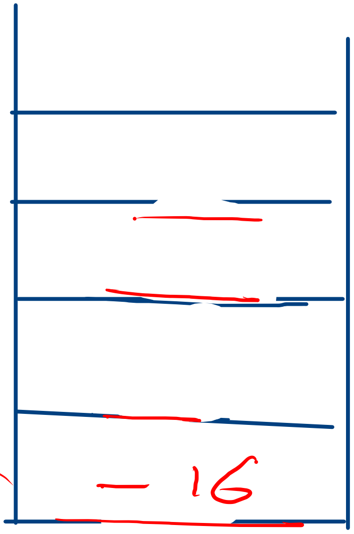
String
4572 + − *

Character (1's Digit

'0' = 48
'1' = 49
'2' = 50

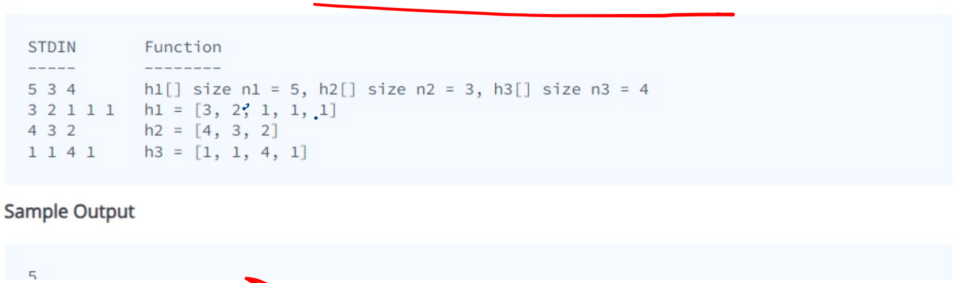
int val1 = −4
int val2 = 4

val2 → val
4 * −4
−16



Sysout(st.pop());

$\underline{3+2+1+1+1} \Rightarrow 8$



```
STDIN          Function
-----          --------
5 3 4          h1[] size n1 = 5, h2[] size n2 = 3, h3[] size n3 = 4
3 2 1 1 1      h1 = [3, 2, 1, 1, 1]
4 3 2          h2 = [4, 3, 2]
1 1 4 1        h3 = [1, 1, 4, 1]
```

Sample Output

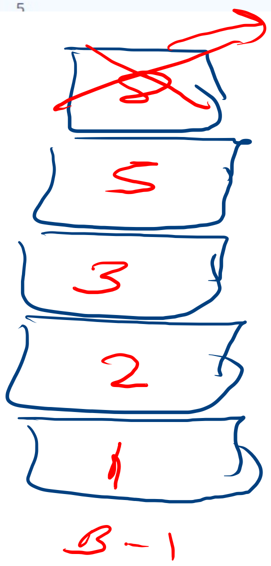
```
5
```

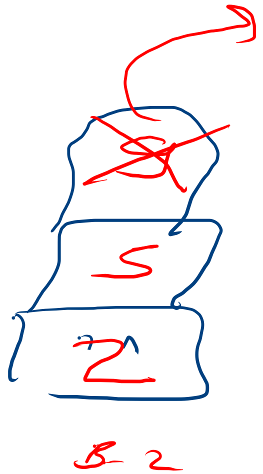
$[3 \quad 2 \quad 1 \quad 1 \quad 1]$

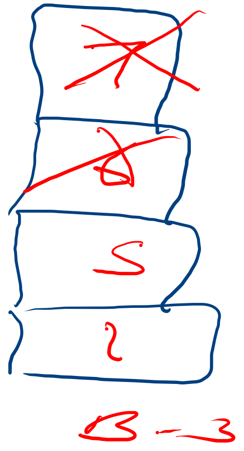
$\{4 \quad 3 \quad 2\}$

$[1, 1, 4, 1]$

$\boxed{5} = \boxed{5}$







s > 0

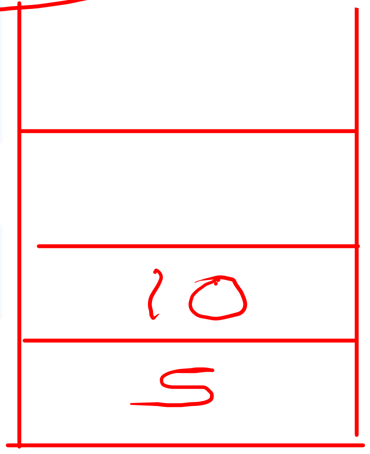
```
3
5
10
~~-5~~
```

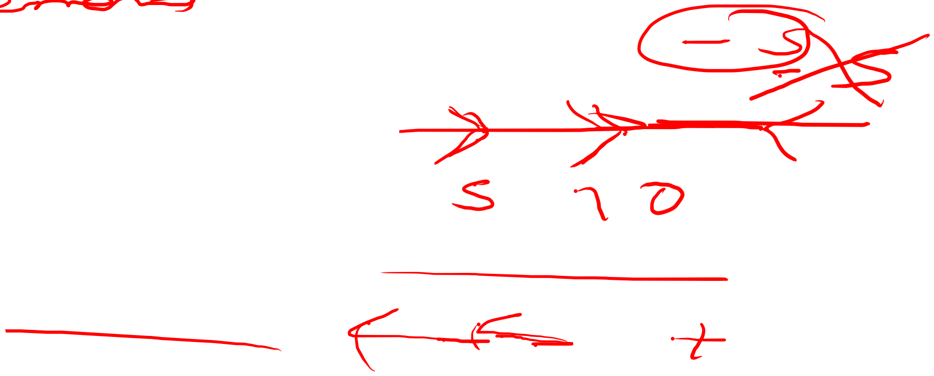
Sample Output 0

```
5 10
```

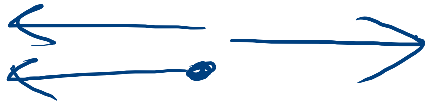
Stack is Empty()

positive element





−5 −10 $\boxed{5}$

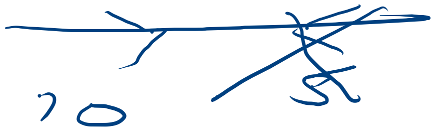


$\boxed{10}$

10

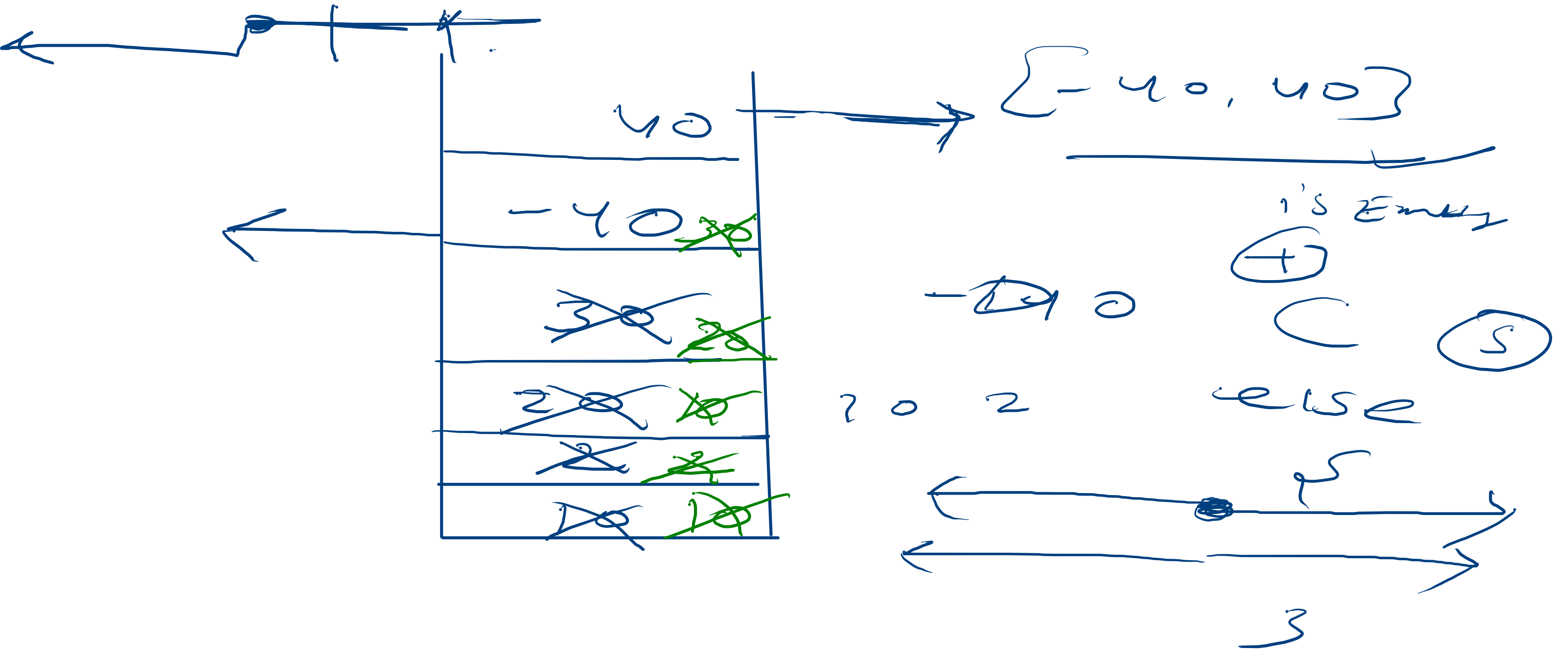
5

```
3
10
2
-5
```

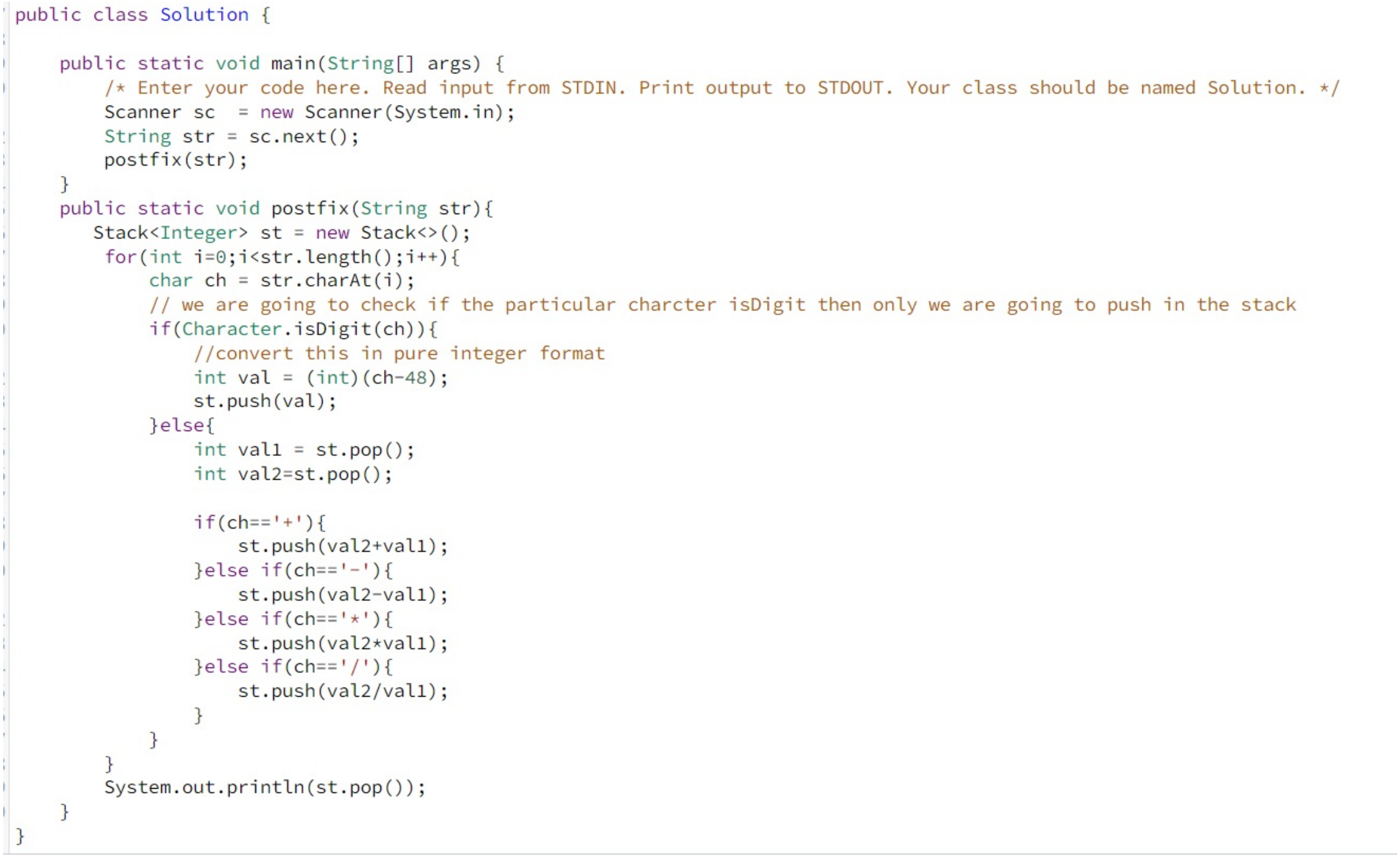


$\boxed{5}$

[10, 2, -5, -10, 20, 30, -40, 40]



postfix expression

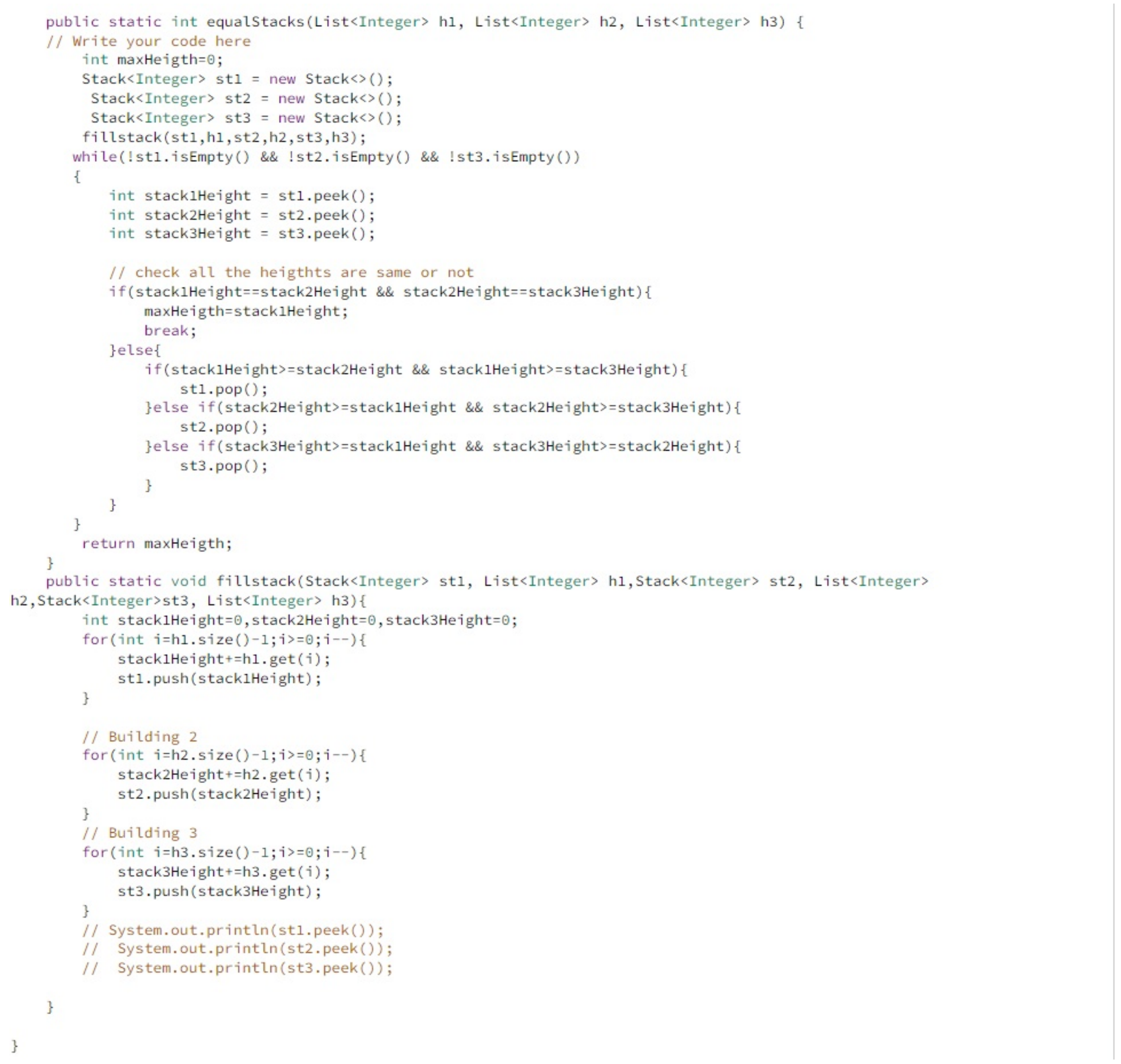


```java
public class Solution {

    public static void main(String[] args) {
        /* Enter your code here. Read input from STDIN. Print output to STDOUT. Your class should be named Solution. */
        Scanner sc  = new Scanner(System.in);
        String str = sc.next();
        postfix(str);
    }
    public static void postfix(String str){
       Stack<Integer> st = new Stack<>();
        for(int i=0;i<str.length();i++){
            char ch = str.charAt(i);
            // we are going to check if the particular charcter isDigit then only we are going to push in the stack
            if(Character.isDigit(ch)){
                //convert this in pure integer format
                int val = (int)(ch-48);
                st.push(val);
            }else{
                int val1 = st.pop();
                int val2=st.pop();

                if(ch=='+'){
                    st.push(val2+val1);
                }else if(ch=='-'){
                    st.push(val2-val1);
                }else if(ch=='*'){
                    st.push(val2*val1);
                }else if(ch=='/'){
                    st.push(val2/val1);
                }
            }
        }
        System.out.println(st.pop());
    }
}
```

Equal stack



```
    public static int equalStacks(List<Integer> h1, List<Integer> h2, List<Integer> h3) {
    // Write your code here
        int maxHeigth=0;
        Stack<Integer> st1 = new Stack<>();
         Stack<Integer> st2 = new Stack<>();
         Stack<Integer> st3 = new Stack<>();
        fillstack(st1,h1,st2,h2,st3,h3);
       while(!st1.isEmpty() && !st2.isEmpty() && !st3.isEmpty())
       {
           int stack1Height = st1.peek();
           int stack2Height = st2.peek();
           int stack3Height = st3.peek();

           // check all the heigthts are same or not
           if(stack1Height==stack2Height && stack2Height==stack3Height){
               maxHeigth=stack1Height;
               break;
           }else{
               if(stack1Height>=stack2Height && stack1Height>=stack3Height){
                   st1.pop();
               }else if(stack2Height>=stack1Height && stack2Height>=stack3Height){
                   st2.pop();
               }else if(stack3Height>=stack1Height && stack3Height>=stack2Height){
                   st3.pop();
               }
           }
       }
        return maxHeigth;
    }
    public static void fillstack(Stack<Integer> st1, List<Integer> h1,Stack<Integer> st2, List<Integer>
h2,Stack<Integer>st3, List<Integer> h3){
        int stack1Height=0,stack2Height=0,stack3Height=0;
        for(int i=h1.size()-1;i>=0;i--){
            stack1Height+=h1.get(i);
            st1.push(stack1Height);
        }

        // Building 2
        for(int i=h2.size()-1;i>=0;i--){
            stack2Height+=h2.get(i);
            st2.push(stack2Height);
        }
        // Building 3
        for(int i=h3.size()-1;i>=0;i--){
            stack3Height+=h3.get(i);
            st3.push(stack3Height);
        }
        // System.out.println(st1.peek());
        //  System.out.println(st2.peek());
        //  System.out.println(st3.peek());

    }

}
```