Postfix expression calculation

infix exp. = ((4+5)*(7-6)) = 9 // (A+B)

prefix exp. = * +45 -76 = 9 // (+AB)

postfix exp. = 45+ 76- * = 9 // (AB+)

Ex infix :- (2/3)*(((7+4)-(3+2)))

1 2 3 4

prefix :- */23 -+74+32

postfix :- 23/ 74+32+- *

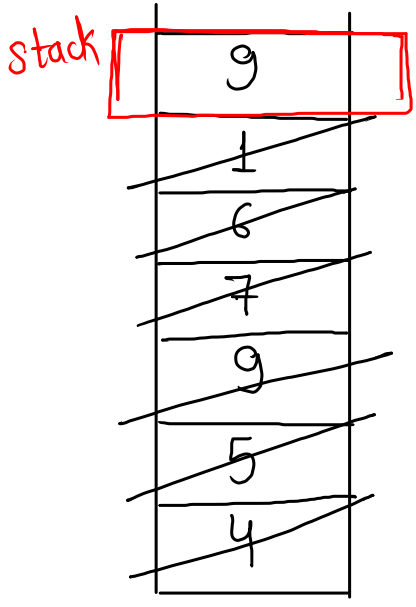
post:

45+ 76- *

↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑

psudo

if got a number
    push
else if got an operator
    get top 2 ele.
    and solve



top1 = ~~5~~ ~~6~~ 1

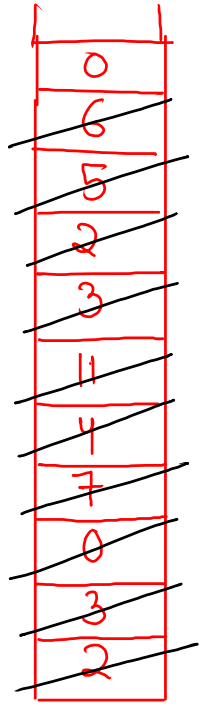
top2 = ~~4~~ ~~7~~ 9

solve = ~~9~~ ~~1~~ 9

solve = top2 - top1

str = "23/74+32+-*"

↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑



ope. = ~~/~~ ~~*~~ ~~*~~ ~~+~~ *

top1 = ~~3~~ ~~4~~ ~~2~~ ~~5~~ 6

top2 = ~~2~~ ~~7~~ ~~3~~ ~~4~~ 0

solve = top2 op top1

= 2/3 = 0

= 7 + 4 = 11

= 3 + 2 = 5

= 11 − 5 = 6

= 0 * 6 = 0

psudo code

1) loop from start to end

1.1) if curr ch is a number
then push in stack

1.2) else if curr ch is an operator

top1 = st.pop()

top2 = st.pop()

solve = top2 operator top1

Imp push solve

Ans = 0

Code

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    String str = scn.nextLine();
    System.out.println(postfixExp(str));
}
public static int postfixExp(String str) {
    Stack<Integer> st = new Stack<>();
    for (int i = 0; i < str.length(); i++) {
        char ch = str.charAt(i);
        if ( Character.isDigit(ch) ) {
            st.push( (ch - '0') );
        } else {
            int top1 = st.pop();
            int top2 = st.pop();
            int solve = 0;
            if ( ch == '+' ) {
                solve = top2 + top1;
            } else if ( ch == '-' ) {
                solve = top2 - top1;
            } else if ( ch == '*' ) {
                solve = top2 * top1;
            } else if ( ch == '/' ) {
                solve = top2 / top1;
            }
            st.push(solve);
        }
    }
    return st.peek();
}
```

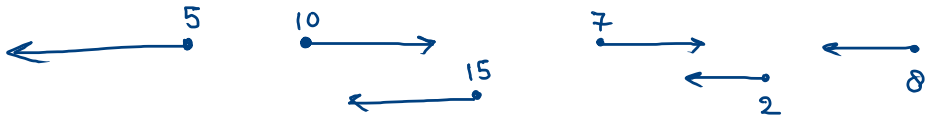
Gm

M. Imp

Asteroid Collision (all have same speed)

arr =

-5	10	-15	7	-2	-8



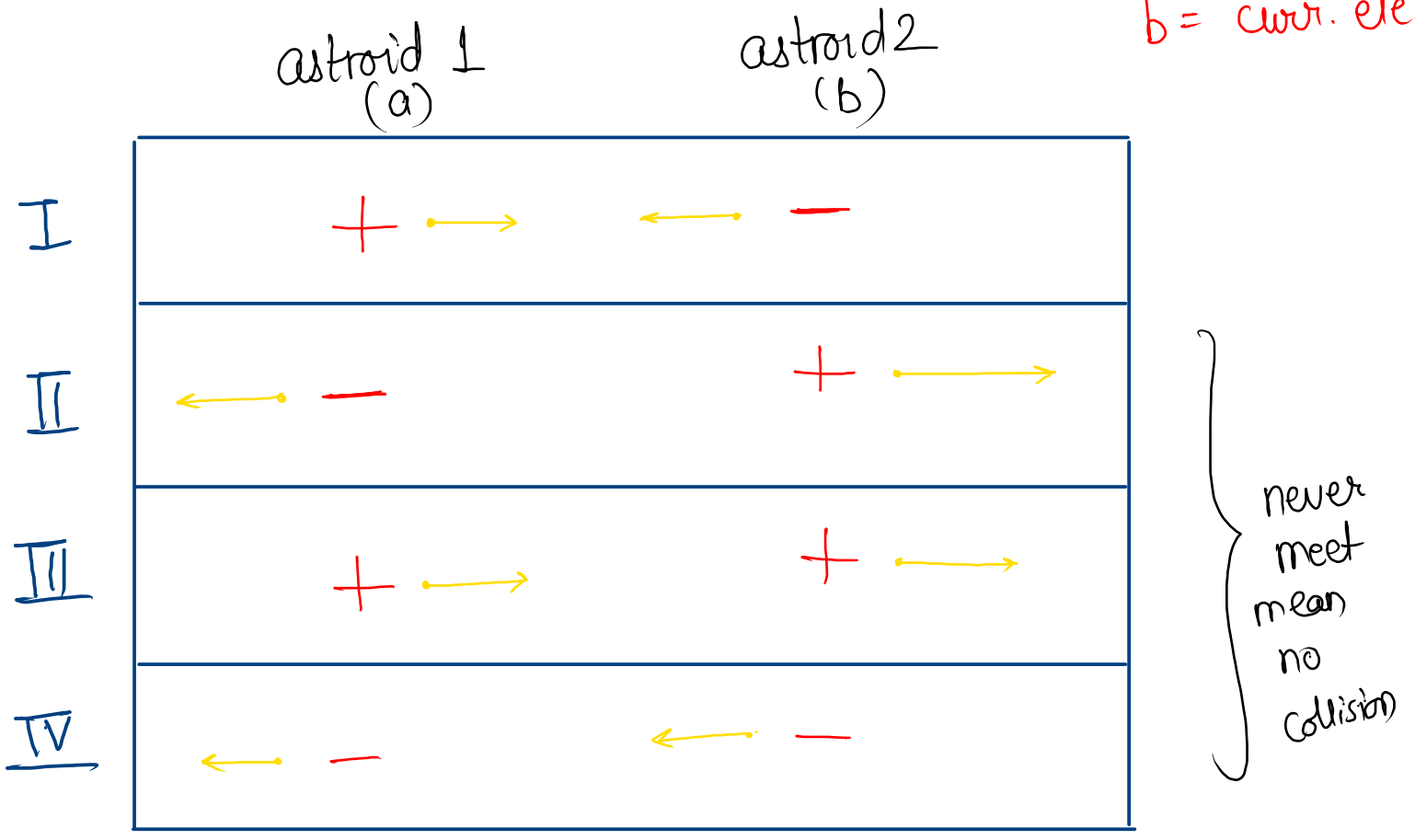
Note:-

- absolute value will be representing size
- + means astroid is moving to right side
- - means astroid is moving to left side

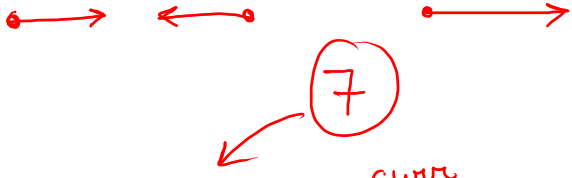
Approach

a = top of st

b = curr. ele.



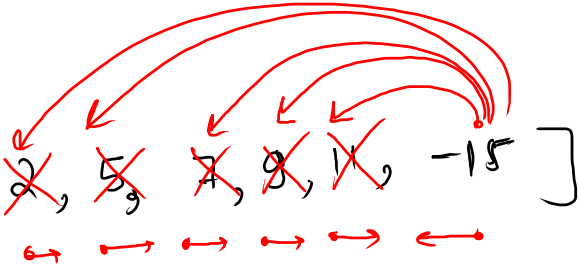
arr = [10 -15 7] = (-15 7)



while(top < (-1)*curr)

pop

ex:- [2, 5, 7, 9, 11, -15]



[-15]

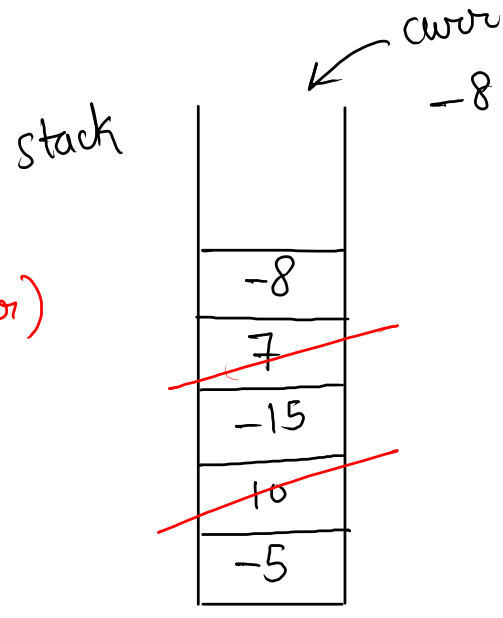
Ex:-

arr =

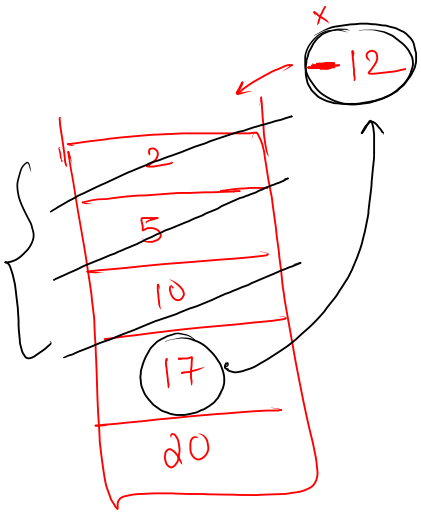
-5	10	-15	7	-2	-8

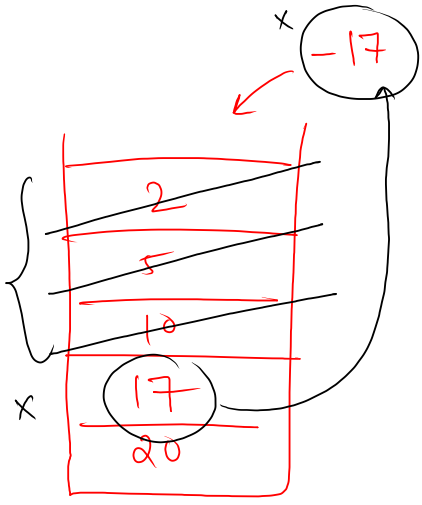
collision scenario :- curr = -ve, top = +ve

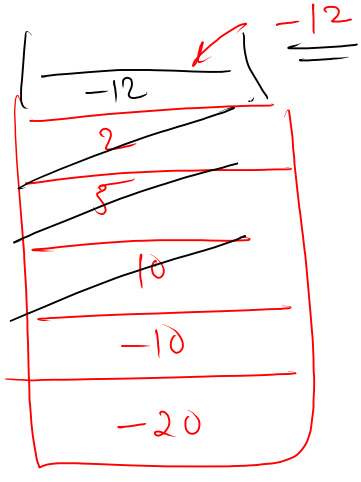
while(top < -1 * curr)
pop()

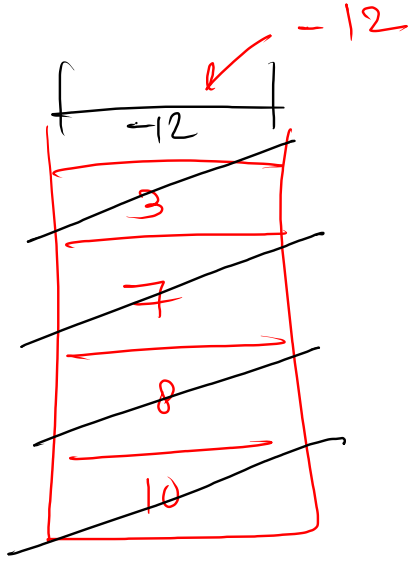


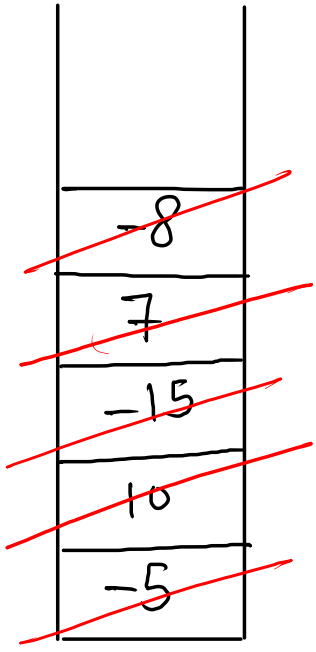
ans = [-5, -15, -8]


x
-12
2
5
10
17
20


x
-17
2
5
10
x
17
20


-12
-12
2
5
10
-10
-20


-12
-12
3
7
8
10



AL

| -8 |

| -15 | -8 |

| -5 | -15 | -8 |

(-5, -15, -8)

Code

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int[] arr = new int[n];
    for (int i = 0; i < n; i++) {
        arr[i] = scn.nextInt();
    }
    ArrayList<Integer> ans = astroidCollision(arr, n);
    for (int i : ans) {
        System.out.print(i + " ");
    }
}
public static ArrayList<Integer> astroidCollision(int[] arr, int n) {
    Stack<Integer> st = new Stack<>();
    for (int i = 0; i < n; i++) {
        if ( arr[i] > 0 ) {
            st.push( arr[i] );
        } else {
            while ( st.size() > 0 && st.peek() > 0 && st.peek() < -1 * arr[i] ) {
                st.pop();
            }
            if ( st.size() > 0 && st.peek() == -1 * arr[i] ) {
                st.pop();
            } else if ( st.isEmpty() || st.peek() < 0 ) {
                st.push( arr[i] );
            }
        }
    }
    ArrayList<Integer> ans = new ArrayList<>();
    while ( st.size() > 0 ) {
        int ele = st.pop();
        ans.add( 0, ele );
    }
    return ans;
}
```