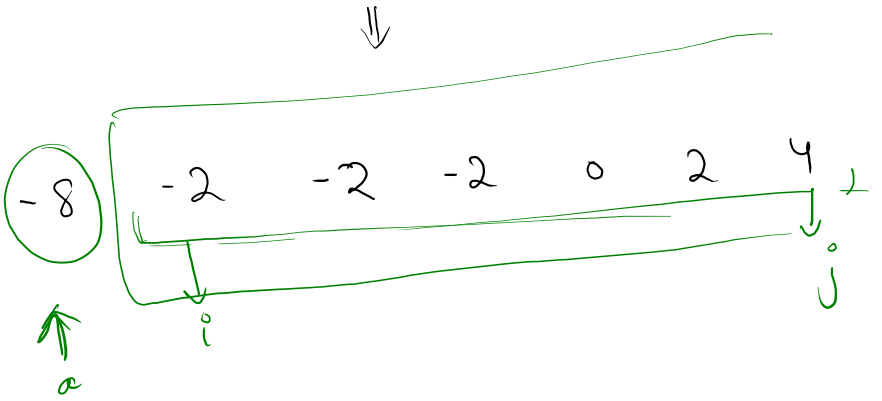
3 sum.

```
6
-2 0 2 4 -2 -8
```

Sample Output 0

```
-2 -2 4
-2 0 2
```

-2 -2 0 2 4 -2 -8



sort

sum = 2

tar : 8

3 sum.

Take an integer array arr as input and print all the triplets `[arr[i], arr[j], arr[k]]` such that `i != j, i != k, and j != k,` and `arr[i] + arr[j] + arr[k] == 0.`

Notice that the solution set must not contain duplicate triplets.

Sample Input 0

```
6
-2 0 2 4 -2 -8
```

i != j

i != k j != k

Sample Output 0

```
-2 -2 4
-2 0 2
```

-2 -2 4 = 0

-2 0 2 = 0

Sample Input 0

```
6
-2 0 2 4 -2 -8
```

Sample Output 0

```
-2 -2 4
-2 0 2
```

tar = 0

-8	-2	-2	0	2	4
0	1	2	3	4	5
a	i				j

$a + b + c = 0$

$b + c = 0 - a$

$\boxed{b + c = -a}$

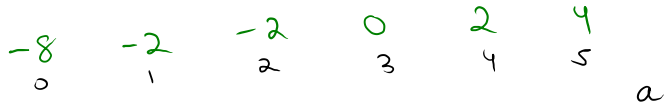
$-8 + b + c = 0$

$b + c = 8$

fix	new tar
-8	8
-2	2
0	0
2	-2
4	-4


```java
//logic
Arrays.sort(A);
for(int a = 0;a < n; a++){
    if(a != 0 && A[a] == A[a-1]){
        continue;
    }
    int i = a+1;
    int j = n-1;
    int nTar = 0-A[a];

    while(i < j){
        int sum  = A[i] + A[j];
        if(sum == nTar){
            System.out.println(A[a] + " " + A[i] + " " + A[j]);
            i++;
            j--;
            while(i < j && A[i] == A[i-1]){ //?
                i++;
            }
            while(i < j && A[j] == A[j+1]){ //?
                j--;
            }
        }else if(sum < nTar){
            i++;
        }else{ //sum > nTar
            j--;
        }
    }
```



ntar = 2

sum = 6

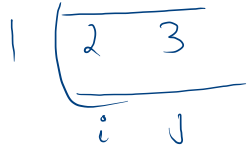
i = 6

j = 5

65

tar = 4.

eg.



Four Sum.

tar := t

new tar
t - A[a] - A[b]

1 1 2 2 3 3 5 5

a b i

Sample Input 0

```
8
3 3 5 5 1 1 2 2
11
```

Sample Output 0

```
1 2 3 5
```

Greatest Till Me

Problem | Submissions | Leaderboard | Discussions

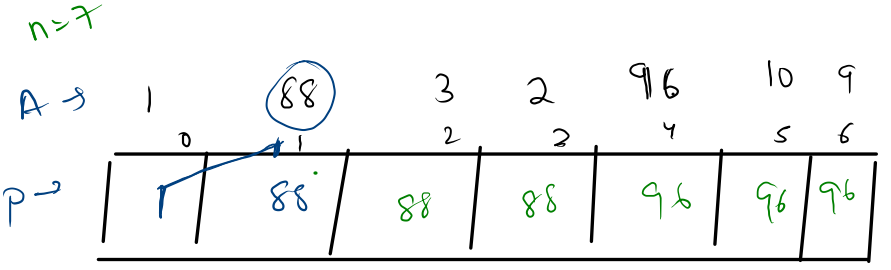
Make a prefix array of size **N** such that at the **kth** index of the prefix array store the greatest element from the left till the **kth** index of the given array.

Sample Input 0

```
7
1
88
3
2
16
10
9
```

Sample Output 0

```
1
88
88
88
88
88
88
```



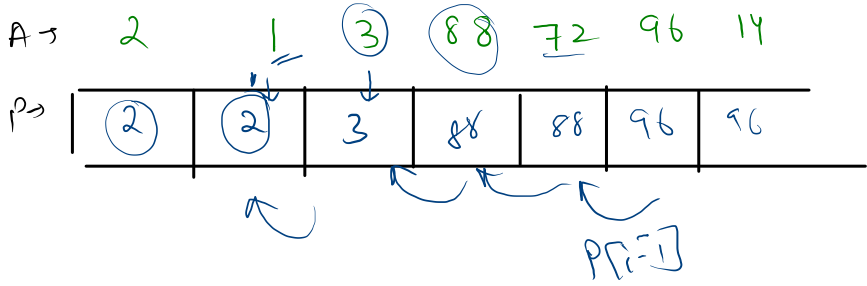
P[0] = A[0]

```java
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int [] A = new int[n];
        for(int i = 0; i < n; i++){
            A[i] = scn.nextInt();
        }
        //logic
        int [] P = new int[n];
        P[0] = A[0];
        for(int i = 1; i < n; i++){
            P[i] = Math.max(A[i], P[i-1]);
        }

        //print
        for(int i = 0; i < n; i++){
            System.out.println(P[i]);
        }
    }
}
```

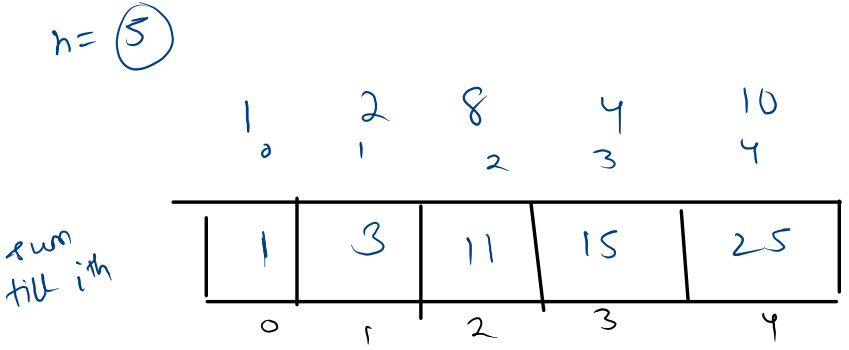


Print Prefix Sum between L and R

Take an integer input **l** and **r** such that `l,r<=array.length`. Given an array. Make a prefix sum array from this. The print the sum of the elements inside the array starting from the **l-index** till the **r-index(l and r both inclusive)**.

Sample Input 0

```
5
1
2
8
4
10
1
3
```



Sample Output 0

```
3
11
15
```

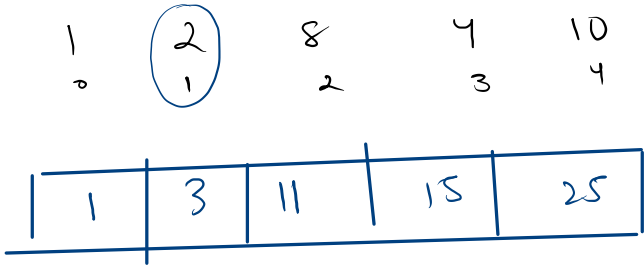
```java
public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int [] A = new int[n];
        for(int i = 0; i < n; i++){
            A[i] = scn.nextInt();
        }
        int l = scn.nextInt();
        int r = scn.nextInt();

        //logic
        int [] P = new int[n];
        P[0] = A[0];
        for(int i = 1; i < n; i++){
            P[i] = A[i] + P[i-1];
        }

        //print
        for(int i = l; i <= r; i++){
            System.out.println(P[i]);
        }
    }
}
```

l=1
r=3



int → 7

Char ch = '7'

ch - '0'

char

int

0 → 'a'

1 → 'b'

= (char) ('a' + 1)