minimum digits

Given an array of digits (values are from 0 to 9), find the **minimum possible sum of two numbers** formed from digits of the array. All digits of the given array must be used to form the two numbers.

Any combination of digits may be used to form the two numbers to be summed. Leading zeroes are permitted.

If forming two numbers is **impossible** (i.e. when n==0) then the "sum" is the value of the only number that can be formed.

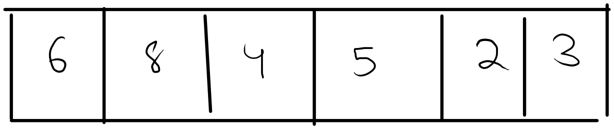
Sample Input 0

```
6
6 8 4 5 2 3
```

Sample Output 0

```
604
```

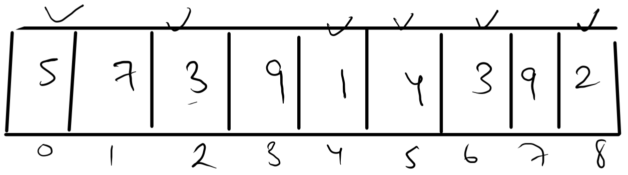
eg.



$$\begin{array}{r} 246 \\ 358 \\ \hline 604 \end{array}$$

$x \to 1\ 3\ 4\ 7\ 9$

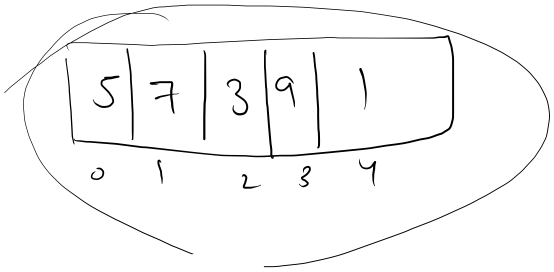
$y \to 2\ 3\ 5\ 9$



$x = 159$

$y = 37$

$\boxed{196}$

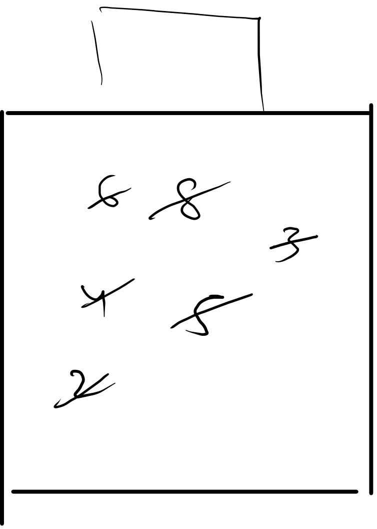


$x = 159$

$y = 37$

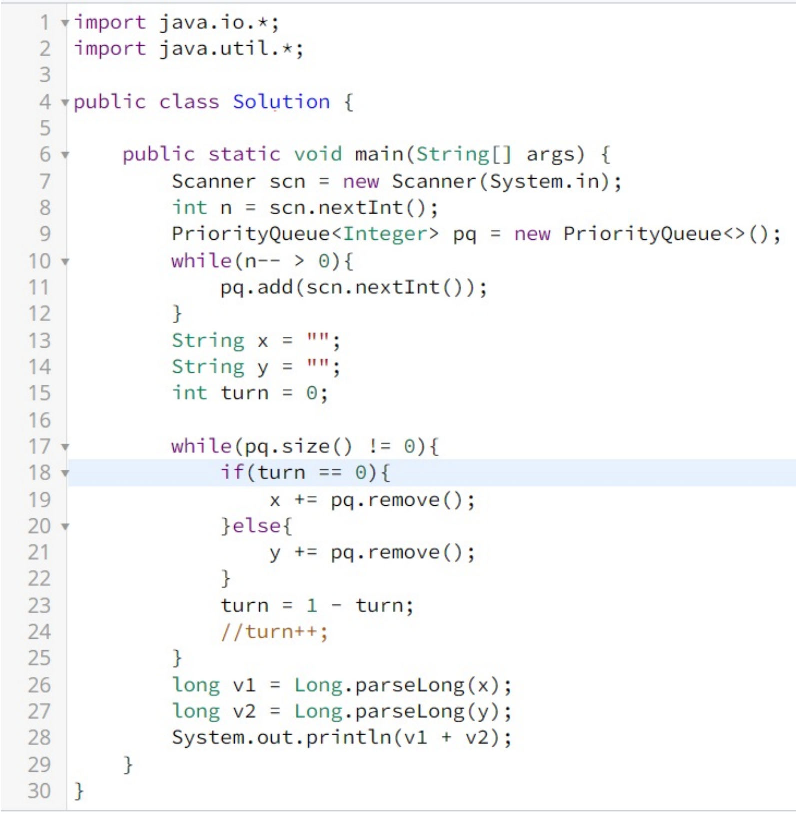
$\boxed{196}$

6 8 4 5 2 3

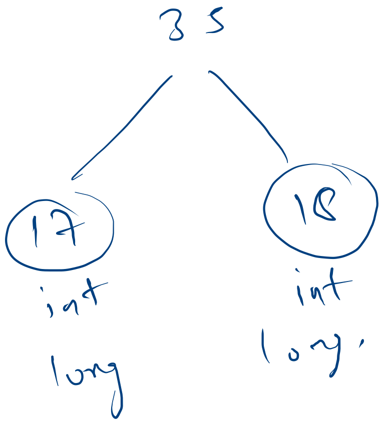


$x = $ "246" $\longrightarrow$ int

$y = $ "358" $\longrightarrow$ int



```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();
        PriorityQueue<Integer> pq = new PriorityQueue<>();
        while(n-- > 0){
            pq.add(scn.nextInt());
        }
        String x = "";
        String y = "";
        int turn = 0;

        while(pq.size() != 0){
            if(turn == 0){
                x += pq.remove();
            }else{
                y += pq.remove();
            }
            turn = 1 - turn;
            //turn++;
        }
        long v1 = Long.parseLong(x);
        long v2 = Long.parseLong(y);
        System.out.println(v1 + v2);
    }
}
```



Merge K sorted arrays

Given k different arrays, which are sorted individually (in ascending order). You need to merge all the given arrays such that output array should be sorted (in ascending order).

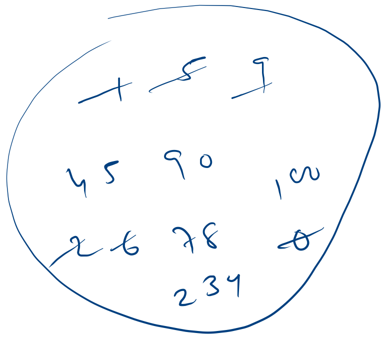
Hint : Use Heaps.

Sample Input 0

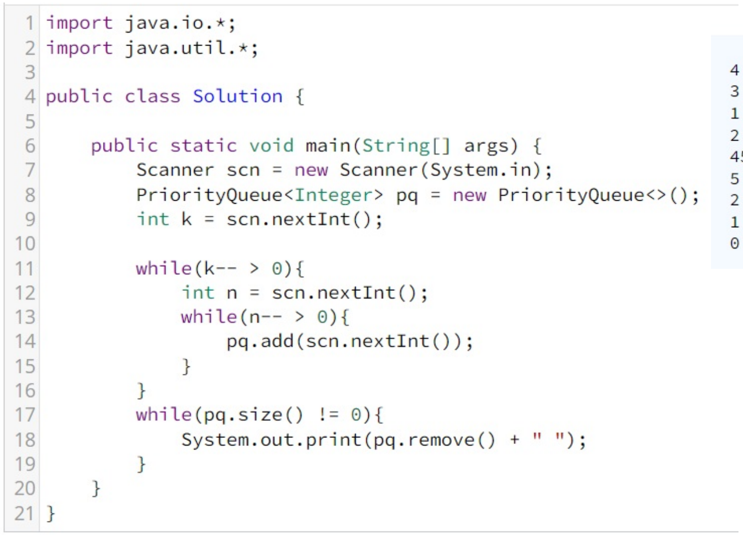
```
4  K
3  → s1
1 5 9
2  → s2
45 90
5  → s3
2 6 78 100 234
1
0
```

Sample Output 0

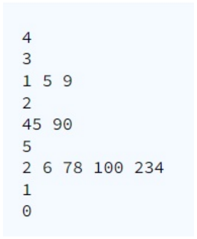
```
0 1 2 5 6 9 45 78 90 100 234
```



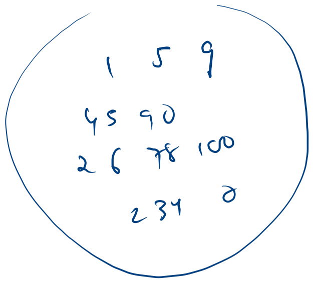
0 1 2 5 6 9....



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

public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        PriorityQueue<Integer> pq = new PriorityQueue<>();
        int k = scn.nextInt();

        while(k-- > 0){
            int n = scn.nextInt();
            while(n-- > 0){
                pq.add(scn.nextInt());
            }
        }
        while(pq.size() != 0){
            System.out.print(pq.remove() + " ");
        }
    }
}
```



```
4
3
1 5 9
2
45 90
5
2 6 78 100 234
1
0
```



weakest rows

You are given an m x n binary matrix **mat** of $1's$ (representing soldiers) and $0's$ (representing civilians). The soldiers are positioned **in front** of the civilians. That is, all the $1's$ will appear to the **left** of all the $0's$ in each row.

A row i is **weaker** than a row j if one of the following is true:

- The number of soldiers in row i is less than the number of soldiers in row j.
- ✓ Both rows have the same number of soldiers and $i < j$.

*Return the indices of the k **weakest** rows in the matrix ordered from weakest to strongest.*

weakest

$\downarrow$

strongest

2
0
3

Sample Input 0

```
5 5 3  ✓→k
1 1 0 0 0
1 1 1 1 0
1 0 0 0 0
1 1 0 0 0
1 1 1 1 1
```

✓
$k = 3$

$1 \rightarrow$ soldier

$0 \rightarrow$ civilian.

$$\begin{array}{c|ccccc|c}
 & & & & & & S \\
0 & 1 & 1 & 0 & 0 & 0 & 2 \\
\hline
1 & 1 & 1 & 1 & 1 & 0 & 4 \\
\hline
2 & 1 & 0 & 0 & 0 & 0 & 1 \\
\hline
3 & 1 & 1 & 0 & 0 & 0 & 2 \\
\hline
4 & 1 & 1 & 1 & 1 & 1 & 5
\end{array}$$

5x5

Sample Output 0

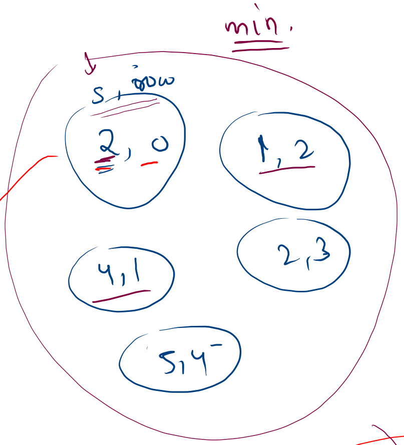
```
2 0 3
```

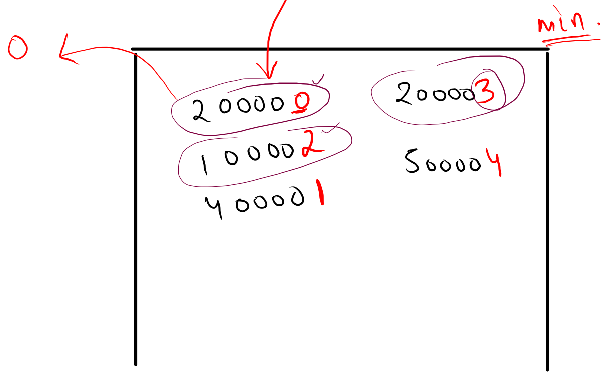
Constraints

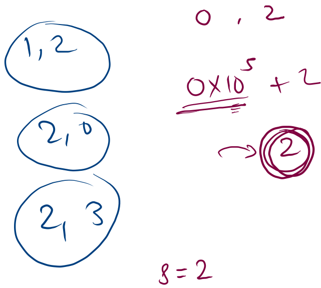
```
m == mat.length
n == mat[i].length
2 <= n, m <= 10000
1 <= k <= m
matrix[i][j] is either 0 or 1.
```

						s
0	1	1	0	0	0	2
1	1	1	1	1	0	4
2	1	0	0	0	0	1
3	1	1	0	0	0	2
4	1	1	1	1	1	5

5 x 5







$0 \times 10^5 + 2 \rightarrow 2$

$s = 2$

add.

$s \times 10^5 + i$

$\cancel{s \times 100 + i}$

100002

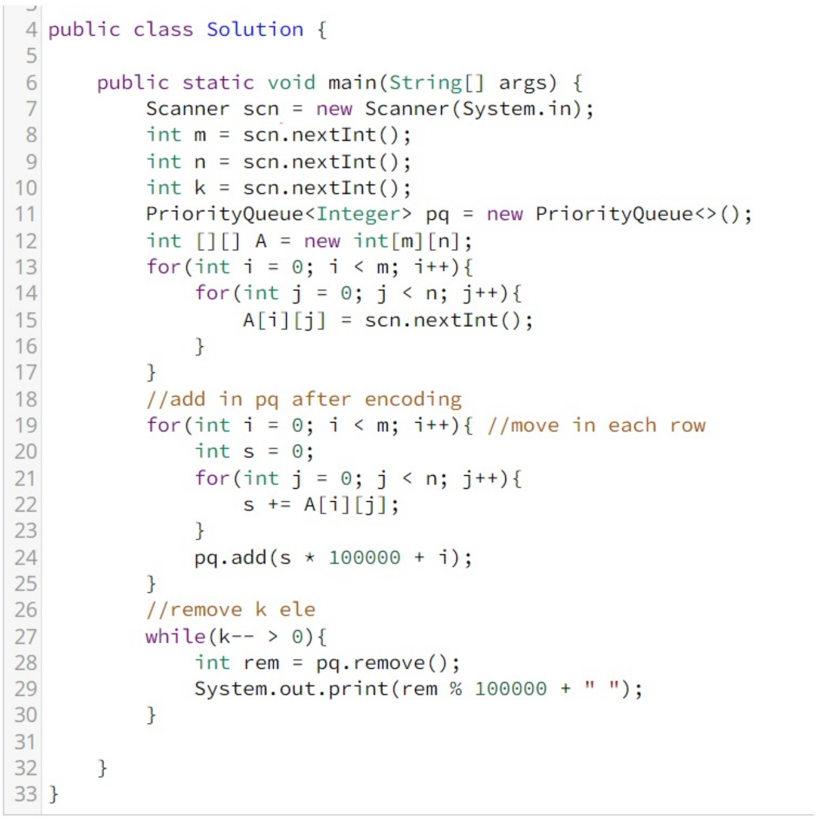
200000

200003

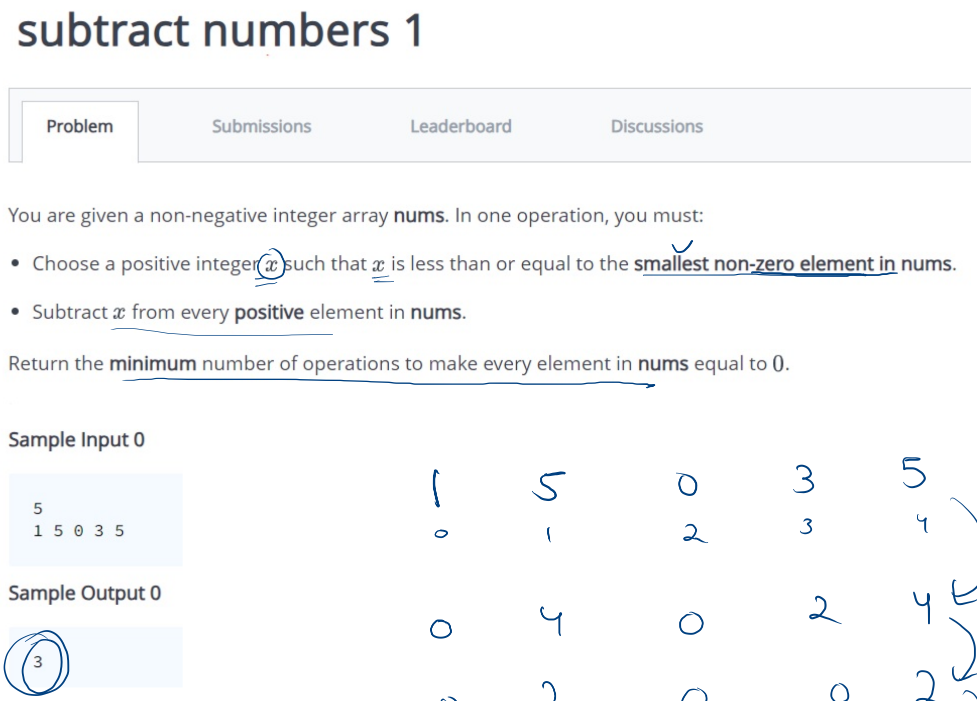
$\boxed{100002 \% 10^5} = 2$

$200000 \% 10^5 = 0$

$200003 \% 10^5 = 3$



```java
public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int m = scn.nextInt();
        int n = scn.nextInt();
        int k = scn.nextInt();
        PriorityQueue<Integer> pq = new PriorityQueue<>();
        int [][] A = new int[m][n];
        for(int i = 0; i < m; i++){
            for(int j = 0; j < n; j++){
                A[i][j] = scn.nextInt();
            }
        }
        //add in pq after encoding
        for(int i = 0; i < m; i++){ //move in each row
            int s = 0;
            for(int j = 0; j < n; j++){
                s += A[i][j];
            }
            pq.add(s * 100000 + i);
        }
        //remove k ele
        while(k-- > 0){
            int rem = pq.remove();
            System.out.print(rem % 100000 + " ");
        }

    }
}
```



subtract numbers 1

You are given a non-negative integer array **nums**. In one operation, you must:

- Choose a positive integer x such that x is less than or equal to the **smallest non-zero element in nums**.
- Subtract x from every **positive** element in **nums**.

Return the **minimum** number of operations to make every element in **nums** equal to 0.

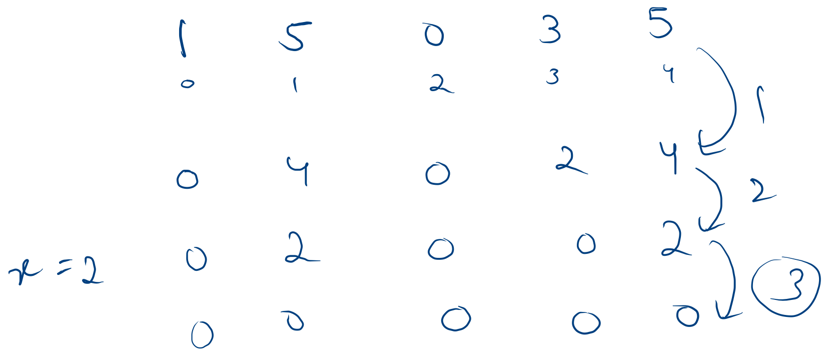
Sample Input 0

```
5
1 5 0 3 5
```

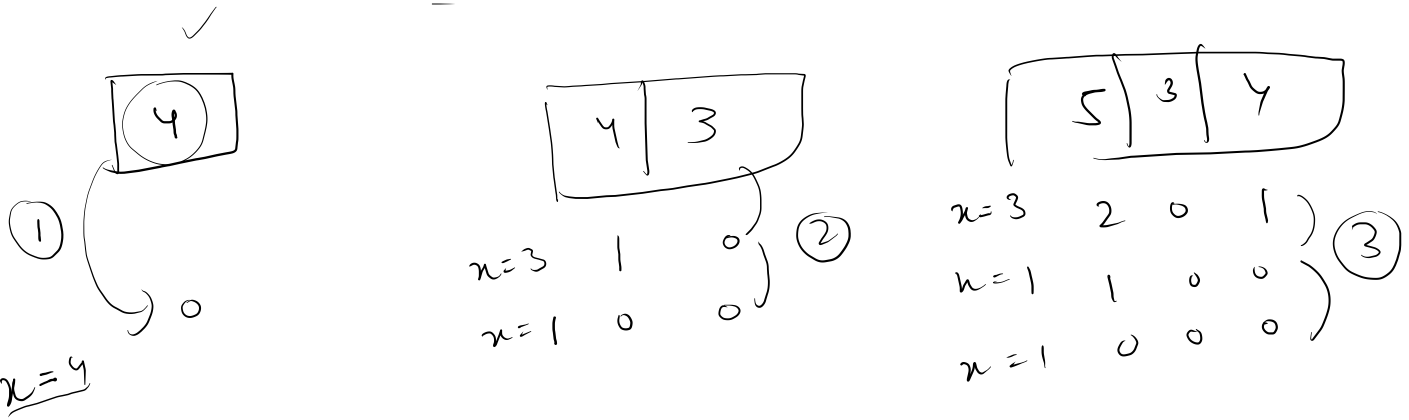
Sample Output 0

```
3
```

+ve 0

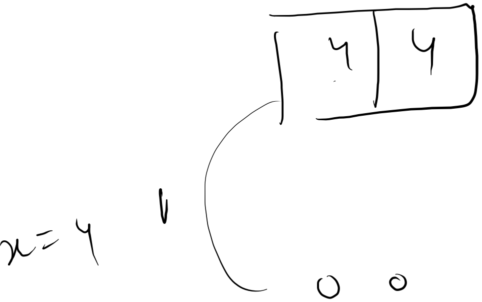


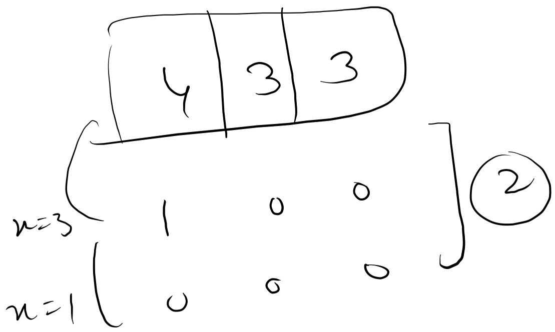
1 5 0 3 5

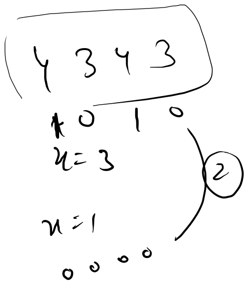


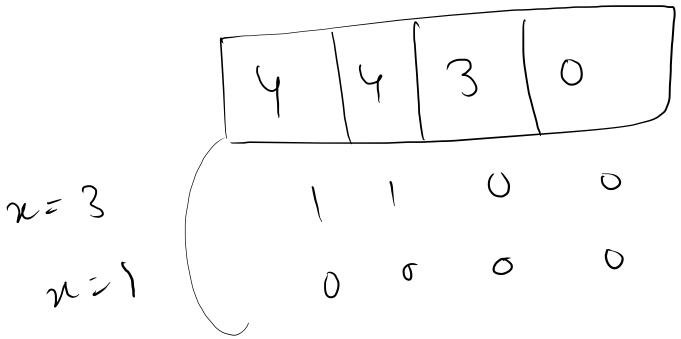
5	3	4	2

$x=2$	3	1	2	0
$x=1$	2	0	1	0
$x=1$	1	0	0	0
$x=1$	0	0	0	0

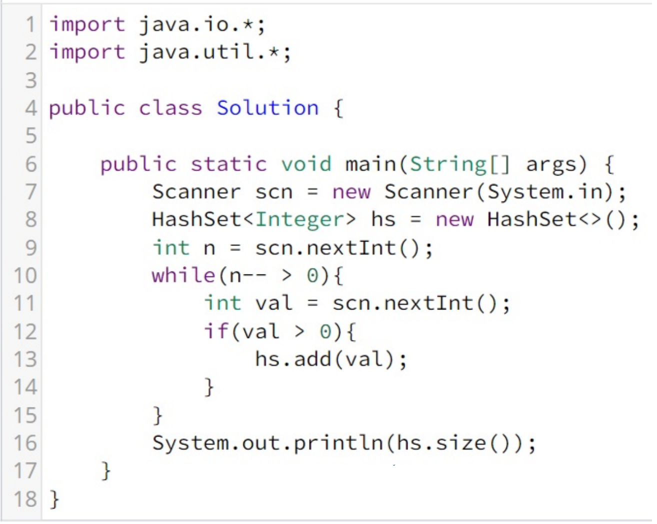








★ no. of unique +ve number



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

public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        HashSet<Integer> hs = new HashSet<>();
        int n = scn.nextInt();
        while(n-- > 0){
            int val = scn.nextInt();
            if(val > 0){
                hs.add(val);
            }
        }
        System.out.println(hs.size());
    }
}
```

n=5	✓ 1	✓ 5	0	✓ 3	✓ 5
	0	1	2	3	4
x=1	0	4	0	2	4
x=2	0	2	0	0	2
					0
x=2	0	0	0	0	

$$\left.\begin{array}{l} TC \to O(n) \\ SC \to O(n) \end{array}\right\}$$