Bubble increasing ____
Decreasing ____

for(________)

for(____) __ __ __ ____

if(arr[i] > arr[i] <

3
2

arr[] = { 1, 2, 5, 6, 8, 9, 12, 10 };

- Array.sort(arr) ;

(nlogn) arr = { 1, 2, 3, 6, 8, 9, 10, 12 }

Custom

Comparable , Comparator

→ implementation

```
public static class myComparator implements Comparator<Integer>{
    @Override
    public int compare(Integer a, Integer b){
        return a-b;
    }
}
```

annotation

logic based on which elements of array will be rearranged

meaning gmp

return a-b; // arrange the elements in ↑ing order

Comparable , Comparator

Comparator

Compare

[2/a, 4/b, 6/b, 8, 7]

a-b

2-4

-ve

axa - bxb increasing

Arrays.sort(arr, new myComparator())

```
public static void main(String[] args) {
    Integer[] arr = {5, -4, 0, -1, 3};
    Arrays.sort(arr, new myComparator());
    for (int i = 0; i < arr.length; i++) {
        System.out.print(arr[i] + " ");
    }
}
public static class myComparator implements Comparator<Integer> {
    @Override
    public int compare(Integer a, Integer b) {
        return a - b; // increasing order
        // return b - a; // decreasing order
    }
}
```

[4/a, 6/b, 2/a, 1/b, 3/a, 5/b]

a-b

b-a

(axa) - (bxb)

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

    Arrays.sort(arr, new myComparator());

    for (int i = 0; i < n; i++) {
        System.out.print(arr[i] + " ");
    }
}
public static class myComparator implements Comparator<Integer> {
    @Override
    public int compare(Integer a, Integer b) {
        return a*a - b*b;
    }
}
```

[4,3,2,8,7,6]

(4x4) - (3x3)

Code

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

    Arrays.sort(arr, (a, b) -> {
        return a*a - b*b;
    });

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

Jmp

lambda

Arrays.sort(arr, (a,b) -> {
return a*a - b*b;
});

Compare (int a, int b){
return a*a - b*b;
}

Mathematics marks of **N** students are arranged in an **array** and two teachers are forming a team each for Maths Olympiad.

They select students turn wise, in each turn, they select a student marks and removes it from the **array**. This goes on until only one mark is left in the array. Considering teacher1 takes the first turn, can you tell us which mark will be left in the array after **N−1** turns.

The first teacher wants to minimize the last number that would be left in the array, while the second teacher wants to maximize it.

You want to know what number will be left in the array after **N−1** turns if both teachers make optimal moves.

Input Format

First line contains **N**, i.e. the size of the array.

Second line contains **N** space-separated positive integers A[i] denoting elements of the array.

Constraints

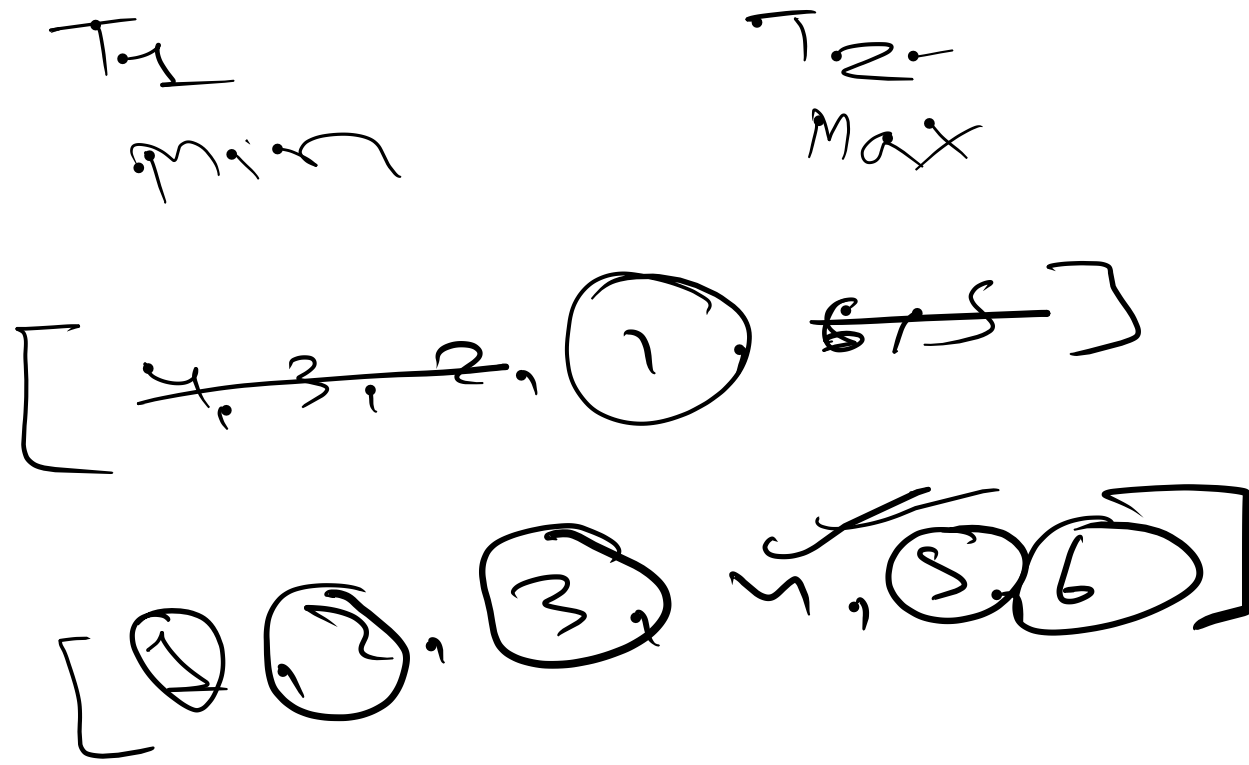
```
1 <= N <= 10^6

1 <= A[i] <= 10^6
```

Output Format

Return the **last** remaining number.

Sample Input 0

```
3
2 1 3
```



Submitted Code

Language: Java 15

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

public class Solution {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int arr[] = new int[n];
        for(int i=0;i<n;i++)arr[i]=sc.nextInt();
        Arrays.sort(arr);

        System.out.print(arr[n/2]);

    }
}
```

Meet Tom, a data analyst who was tasked with analyzing a dataset that contained information about the sales of a retail company. Tom needed to find the **third highest** sales amount from the dataset.

Help Tom to write a program that take an **array** of sales amount as input and return the **third highest** sales amount in the array and If the **third highest** amount does not exist than **return** the **highest amount**.

NOTE :- After answering the question, attempt the related question in the linked resource to improve your understanding of the question. Click here

Input Format

First line of input contains integer **N** representing the size of array.

Second line of input contains **N** integers representing the elements of array.

Constraints

```
1 <= nums.length <= 10^4
-2^31 <= nums[i] <= 2^31 - 1
```

Output Format

Return the **third maximum** element.

Sample Input 0

```
3
3 2 1
```

Sample Output 0

```
1
```

Explanation 0

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

public class Solution {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int arr[] = new int[n];
        for(int i=0;i<n;i++)arr[i]=sc.nextInt();

        System.out.print(thirdMax(arr));
    }
    public static int thirdMax(int [] arr){
        int fMax = Integer.MIN_VALUE;
        int sMax = Integer.MIN_VALUE;
        int tMax = Integer.MIN_VALUE;

        for(int i =0 ;i<arr.length;i++){
            if(fMax <arr[i]){
                tMax=sMax;
                sMax=fMax;
                fMax= arr[i];
            }
            else if( fMax > arr[i] && sMax <arr[i]){
                tMax = sMax;
                sMax = arr[i];
            }
            else if(fMax > arr[i] && sMax > arr[i] && tMax <arr[i]){
                tMax = arr[i];
            }
        }

        if(tMax== Integer.MIN_VALUE)return fMax;
        else return tMax;


    }
}
```

Once upon a time, there was a mathematician named Max who loved solving number puzzles. One day, he was given an array of integers and was challenged to find **the maximum product of three numbers** from the array.

Max eagerly accepted the challenge and began working on it. Help Max to find the maximum product of three numbers.

NOTE:- After answering the question, attempt the related question in the linked resource to improve your understanding of the question . Click here

Input Format

An integer **N**, which is the size of the array.

N integers, depicting the elements of the array.

Constraints

```
- 3<=N<=1000
- -1000<=arr[i]<=1000
```

Output Format

Return the **maximum product of three numbers** from the array.

Sample Input 0

```
5
-7 3 -5 2 4
```

Sample Output 0

```
140
```

Explanation 0

Product of **-7, -5 and 4** will give 140.