Lab: Arrays

Problems for exercises and homework for the "Technology Fundamentals" course @ SoftUni.

You can check your solutions in Judge.

1. Day of Week

Enter a **day number** [1…7] and print the **day name** (in English) or "**Invalid day!**". Use an **array of strings**.

Examples

Input	Output
1	Monday
2	Tuesday
7	Sunday
0	Invalid day!

Hints

- Use an **array of strings** holding the day names: {"Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"}.
- Print the element at index (**day-1**) when it is in the range [1…7] or "**Invalid Day!**" otherwise.

2. Print Numbers in Reverse Order

Read n numbers and print them in reverse order.

Examples

Input	Output
3 10 20 30	30 20 10
3 30 20 10	10 20 30
1 10	10

Solution

First, we need to read **n** from the console.

© SoftUni – https://softuni.org. Copyrighted document. Unauthorized copy, reproduction or use is not permitted.

```
public class PrintNumbersInReversedOrder {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        int n = Integer.parseInt(scanner.nextLine());
    }
}
```

Create an **array of integer** with **n** size.

```
public class PrintNumbersInReversedOrder {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        int n = Integer.parseInt(scanner.nextLine());

        int[] numbers = new int[n];
    }
}
```

Read **n** numbers using for loop and fill the array.

```
for (int i = 0; i < n; i++) {
    int number = Integer.parseInt(scanner.nextLine());
    numbers[i] = number;
}
```

Print the array in reversed order.

```
for (int i = numbers.length - 1; i >= 0; i--) {
    System.out.println(numbers[i]);
}
```

3. Sum Even Numbers

Read an array from the console and sum only the even numbers.

Examples

Input	Output
1 2 3 4 5 6	12
3 5 7 9	0
2 4 6 8 10	30

Solution

First, we need to read the array.

```
int[] numbers = Arrays
        .stream(scanner.nextLine().split( regex: " "))
        .mapToInt(e -> Integer.parseInt(e))
        .toArray();
```

We will need a variable for the sum.

```
int sum = 0;
```


© SoftUni – https://softuni.org. Copyrighted document. Unauthorized copy, reproduction or use is not permitted.

Iterate through all elements in the array with for loop. If the number is even add it to the sum.

```
for (int i = 0; i < numbers.length; i++) {
    if (numbers[i] % 2 == 0) {
        sum += numbers[i];
    }
}
```

Print the total sum

4. Reverse an Array of Strings

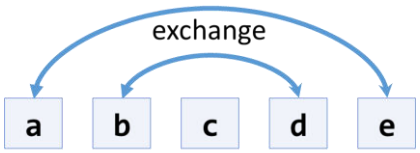
Write a program to read **an array of strings**, **reverse** it and **print** its elements. The input consists of a sequence of space separated strings. Print the output on a single line (space separated).

Examples

Input	Output
a b c d e	e d c b a
-1 hi ho w	w ho hi -1

Hints

- Read the array of strings.
- **Exchange** the **first** element (at index 0) with the **last** element (at index n-1).
- **Exchange** the **second** element (at index 1) with the element **before the last** (at index n-2).
- Continue the same way until the middle of the array is reached.



5. Even and Odd Subtraction

Write a program that calculates the difference between the sum of the even and the sum of the odd numbers in an array.

Examples

Input	Output	Comments
1 2 3 4 5 6	3	2 + 4 + 6 = 12 1 + 3 + 5 = 9 12 – 9 = 3
3 5 7 9	-24	
2 4 6 8 10	30	

Solution

First, we need to read the array.

© SoftUni – https://softuni.org. Copyrighted document. Unauthorized copy, reproduction or use is not permitted.

```
int[] numbers = Arrays
        .stream(scanner.nextLine().split( regex: " "))
        .mapToInt(e -> Integer.parseInt(e))
        .toArray();
```

We will need two variables – even and odd sum.

```
int evenSum = 0;
int oddSum = 0;
```

Iterate through all elements in the array. Check the current number – if it is even add it to the even sum, otherwise add it to the odd sum.

```
for (int number : numbers) {
    if (number % 2 == 0) {
        evenSum += number;
    } else {
        oddSum += number;
    }
}
```

Print the difference.

```
int diff = evenSum - oddSum;
System.out.println(diff);
```

6. Equal Arrays

Read two arrays and print on the console whether they are identical or not. Arrays are identical if their elements are equal. If the arrays are identical find the sum of the first one and print on the console following message: "Arrays are identical. Sum: {sum}", otherwise find the first index where the arrays differ and print on the console following message: "Arrays are not identical. Found difference at {index} index."

Examples

Input	Output
10 20 30 10 20 30	Arrays are identical. Sum: 60
1 2 3 4 5 1 2 4 3 5	Arrays are not identical. Found difference at 2 index.
1 10	Arrays are not identical. Found difference at 0 index.

Hints

First, we need to read two arrays.

© SoftUni – https://softuni.org. Copyrighted document. Unauthorized copy, reproduction or use is not permitted.
Follow us:

```
Scanner scanner = new Scanner(System.in);

int[] firstArr = Arrays
        .stream(scanner.nextLine().split( regex: " "))
        .mapToInt(e -> Integer.parseInt(e))
        .toArray();

int[] secondArr = Arrays
        .stream(scanner.nextLine().split( regex: " "))
        .mapToInt(Integer::parseInt)
        .toArray();
```

Iterate through arrays and compare element. If the elements are not equal print the required message and break the loop.

```
for (int i = 0; i < maxLength; i++) {
    sum+=firstArr[i];
    if (firstArr[i] != secondArr[i]) {
        System.out.printf("Arrays are not identical. Found difference at %d index.", i);
        break;
    }
}
```

Think about how to solve the other part of the problem.

7. Condense Array to Number

Write a program to read **an array of integers** and **condense** them by **summing** adjacent couples of elements until a **single integer** is obtained. For example, if we have 3 elements {2, 10, 3}, we sum the first two and the second two elements and obtain {2+10, 10+3} = {12, 13}, then we sum again all adjacent elements and obtain {12+13} = {25}.

Examples

Input	Output	Comments
2 10 3	25	2 10 3 → 2+10 10+3 → 12 13 → 12 + 13 → 25
5 0 4 1 2	35	5 0 4 1 2 → 5+0 0+4 4+1 1+2 → 5 4 5 3 → 5+4 4+5 5+3 → 9 9 8 → 9+9 9+8 → 18 17 → 18+17 → 35
1	1	1 is already condensed to number

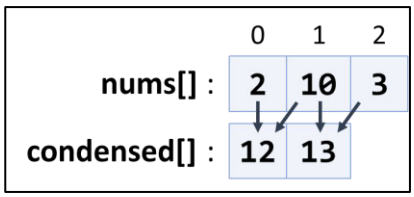
Hints

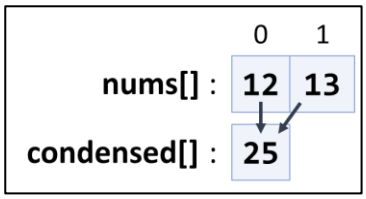
While we have more than one element in the array **nums[]**, repeat the following:

- Allocate a new array **condensed[]** of size **nums.Length-1**.
- Sum the numbers from **nums[]** to **condensed[]**:
    - **condensed[i] = nums[i] + nums[i+1]**
- **nums[] = condensed[]**

The process is illustrated below:

© SoftUni – https://softuni.org. Copyrighted document. Unauthorized copy, reproduction or use is not permitted.
Follow us:





© SoftUni – https://softuni.org. Copyrighted document. Unauthorized copy, reproduction or use is not permitted.

Exercise: Arrays

Problems for exercises and homework for the "Technology Fundamentals" course @ SoftUni.

You can check your solutions in Judge.

1. Train

You will be given a count of wagons in a train **n**. On the next **n** lines, you will receive how many people are going to get on that wagon. At the end print the whole train and after that the sum of the people in the train.

Examples

Input	Output
3 13 24 8	13 24 8 45
6 3 52 71 13 65 4	3 52 71 13 65 4 208
1 100	100 100

2. Common Elements

Write a program, which prints common elements in two arrays. You have to compare the elements of the second array to the elements of the first.

Examples

Input	Output
Hey hello 2 4 10 hey 4 hello	4 hello
S of t un i of i 10 un	of i un
i love to code code i love to	code i love to

3. Zig-Zag Arrays

Write a program which creates 2 arrays. You will be given an integer **n**. On the next **n** lines, you get 2 integers. Form 2 arrays as shown below.

© SoftUni – https://softuni.org. Copyrighted document. Unauthorized copy, reproduction or use is not permitted.

Examples

Input	Output
4 1 5 9 10 31 81 41 20	1 10 31 20 5 9 81 41
2 80 23 31 19	80 19 23 31

4. Array Rotation

Write a program that receives an array and number of rotations you have to perform (first element goes at the end) Print the resulting array.

Examples

Input	Output
51 47 32 61 21 2	32 61 21 51 47
32 21 61 1 4	32 21 61 1
2 4 15 31 5	4 15 31 2

5. Top Integers

Write a program to find all the top integers in an array. A top integer is an integer which is **bigger** than all the elements to its right.

Examples

Input	Output
1 4 3 2	4 3 2
14 24 3 19 15 17	24 19 17
27 19 42 2 13 45 48	48

6. Equal Sums

Write a program that determines if there **exists an element in the array** such that the **sum of the elements on its left** is **equal** to the **sum of the elements on its right**. If there are **no elements to the left / right**, their **sum is considered to be 0**. Print the **index** that satisfies the required condition or **"no"** if there is no such index.

Examples

Input	Output	Comments
1 2 3 3	2	At a[2] -> left sum = 3, right sum = 3

© SoftUni – https://softuni.org. Copyrighted document. Unauthorized copy, reproduction or use is not permitted.
Follow us:

		a[0] + a[1] = a[3]
1 2	no	At a[0] -> left sum = 0, right sum = 2 At a[1] -> left sum = 1, right sum = 0 No such index exists
1	0	At a[0] -> left sum = 0, right sum = 0
1 2 3	no	No such index exists
10 5 5 99 3 4 2 5 1 1 4	3	At a[3] -> left sum = 20, right sum = 20 a[0] + a[1] + a[2] = a[4] + a[5] + a[6] + a[7] + a[8] + a[9] + a[10]

7. Max Sequence of Equal Elements

Write a program that finds the **longest sequence of equal elements** in an array of integers. If several longest sequences exist, print the leftmost one.

Examples

Input	Output
2 1 1 2 3 3 **2 2 2** 1	2 2 2
1 1 1 2 3 1 3 3	1 1 1
4 4 4 4	4 4 4 4
0 **1 1** 5 2 2 6 3 3	1 1

8. Magic Sum

Write a program, which prints all unique pairs in an array of integers whose sum is equal to a given number.

Examples

Input	Output
1 7 6 2 19 23 8	1 7 6 2
14 20 60 13 7 19 8 27	14 13 20 7 19 8

9. *Kamino Factory

The clone factory in Kamino got another order to clone troops. But this time you are tasked to find **the best DNA** sequence to use in the production.

You will receive the **DNA length** and until you receive the command **"Clone them!"** you will be receiving a **DNA sequences of ones and zeroes, split by "!" (one or several).**

You should select the sequence with the **longest subsequence of ones**. If there are several sequences with **same length of subsequence of ones**, print the one with the **leftmost starting index**, if there are several sequences with same **length and starting index**, select the sequence with the **greater sum** of its elements.

After you receive the last command **"Clone them!"** you should print the collected information in the following format:

© SoftUni – https://softuni.org. Copyrighted document. Unauthorized copy, reproduction or use is not permitted.

"Best DNA sample {bestSequenceIndex} with sum: {bestSequenceSum}."

"{DNA sequence, joined by space}"

Input / Constraints

- The **first line** holds the **length** of the **sequences** – **integer in range [1…100];**
- On the next lines until you receive **"Clone them!"** you will be receiving sequences (at least one) of ones and zeroes, **split by "!"** (one or several).

Output

The output should be printed on the console and consists of two lines in the following format:

"Best DNA sample {bestSequenceIndex} with sum: {bestSequenceSum}."

"{DNA sequence, joined by space}"

Examples

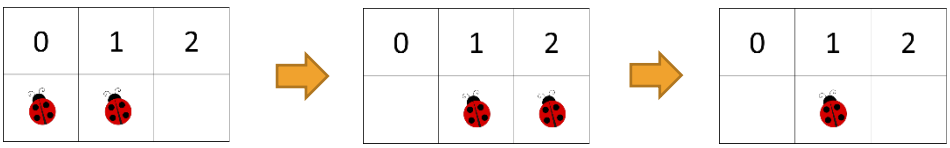
Input	Output	Comments
5 1!0!1!1!0 0!1!1!0!0 Clone them!	Best DNA sample 2 with sum: 2. 0 1 1 0 0	We receive 2 sequences with **same length of subsequence of ones**, but the second is printed, because its subsequence starts at **index[1].**
Input	**Output**	**Comments**
4 1!1!0!1 1!0!0!1 1!1!0!0 Clone them!	Best DNA sample 1 with sum: 3. 1 1 0 1	We receive 3 sequences. Both 1 and 3 **have same length** of subsequence of ones -> 2, **and both start from index[0]**, but the first is printed, because its **sum is greater.**

10. *LadyBugs

You are **given a field size** and the **indexes of ladybugs** inside the field. After that on every new line **until the "end" command** is given, a **ladybug changes its position** either to its **left or to its right by a given fly length**.

A **command to a ladybug** looks like this: "**0 right 1**". This means that the little insect placed on index 0 should fly one index to its right. If the ladybug **lands on a fellow ladybug**, it **continues to fly** in the same direction **by the same fly length**. If the ladybug **flies out of the field, it is gone**.

For example, imagine you are given a field with size 3 and ladybugs on indexes 0 and 1. If the ladybug on index 0 needs to fly to its right by the length of 1 (0 right 1) it will attempt to land on index 1 but as there is another ladybug there it will continue further to the right by additional length of 1, landing on index 2. After that, if the same ladybug needs to fly to its right by the length of 1 (2 right 1), it will land somewhere outside of the field, so it flies away:



If you are given ladybug index that does not have ladybug there, nothing happens. If you are given ladybug index that is outside the field, nothing happens.

© SoftUni – https://softuni.org. Copyrighted document. Unauthorized copy, reproduction or use is not permitted.
Follow us:

Your job is to create the program, simulating the ladybugs flying around doing nothing. At the end, **print all cells in the field separated by blank spaces**. For each cell that has a ladybug on it print '**1**' and for each empty cells print '**0**'. For the example above, the output should be **'0 1 0'**.

Input

- On the first line you will receive an integer - the size of the field
- On the second line you will receive the initial **indexes** of all ladybugs separated by a blank space. **The given indexes** may or may not be inside the field range
- On the next lines, until you get the "end" command you will receive commands in the format: "**{ladybug index} {direction} {fly length}**"

Output

- Print the **all cells within the field in format: "{cell} {cell} … {cell}"**
  - If a cell has ladybug in it, print **'1'**
  - If a cell is empty, print **'0'**

Constrains

- The size of the field will be in the range [0 … 1000]
- The ladybug indexes will be in the range [-2,147,483,647 … 2,147,483,647]
- The number of commands will be in the range [0 … 100]
- The fly length will be in the range [-2,147,483,647 … 2,147,483,647]

Examples

Input	Output	Comments
3 0 1 0 right 1 2 right 1 end	0 1 0	1 1 0 - Initial field 0 1 1 - field after "0 right 1" 0 1 0 - field after "2 right 1"

Input	Output
3 0 1 2 0 right 1 1 right 1 2 right 1 end	0 0 0
5 3 3 left 2 1 left -2 end	0 0 0 1 0

© SoftUni – https://softuni.org. Copyrighted document. Unauthorized copy, reproduction or use is not permitted.

More Exercise: Arrays

Problems for exercises and homework for the "Technology Fundamentals" course @ SoftUni.

You can check your solutions in Judge.

1. Encrypt, Sort and Print Array

Write a program that reads a **sequence of strings** from the console. Encrypt every string by summing:

- The code of **each vowel multiplied by the string length**
- The code of **each consonant divided by the string length**

Sort the **number** sequence in ascending order and print it on the console.

On first line, you will always receive the number of strings you have to read.

Examples

Input	Output	Comments
4 Peter Maria Katya Todor	1032 1071 1168 1532	Peter = 1071 Maria = 1532 Katya = 1032 Todor = 1168
3 Sofia London Washington	1396 1601 3202	Sofia = 1601 London = 1396 Washington = 3202

2. Pascal Triangle

The triangle may be constructed in the following manner: In row 0 (the topmost row), there is a unique nonzero entry 1. Each entry of each subsequent row is constructed by adding the number above and to the left with the number above and to the right, treating blank entries as 0. For example, the initial number in the first (or any other) row is 1 (the sum of 0 and 1), whereas the numbers 1 and 3 in the third row are added to produce the number 4 in the fourth row.

If you want more info about it: https://en.wikipedia.org/wiki/Pascal's_triangle

Print each row elements separated with whitespace.

Examples

Input	Output
4	1 1 1 1 2 1 1 3 3 1
13	1 1 1 1 2 1 1 3 3 1 1 4 6 4 1

© SoftUni – https://softuni.org. Copyrighted document. Unauthorized copy, reproduction or use is not permitted.

	1 5 10 10 5 1 1 6 15 20 15 6 1 1 7 21 35 35 21 7 1 1 8 28 56 70 56 28 8 1 1 9 36 84 126 126 84 36 9 1 1 10 45 120 210 252 210 120 45 10 1 1 11 55 165 330 462 462 330 165 55 11 1 1 12 66 220 495 792 924 792 495 220 66 12 1

Hints

- The input number **n** will be **1 <= n <= 60**
- Think about proper **type** for elements in array
- Don't be scary to use **more and more arrays**

3. Recursive Fibonacci

The Fibonacci sequence is quite a famous sequence of numbers. Each member of the sequence is calculated from the sum of the two previous elements. The **first two** elements are 1, 1. Therefore the sequence goes as 1, 1, 2, 3, 5, 8, 13, 21, 34…
The following sequence can be generated with an array, but that's easy, so your task is to implement recursively.

So if the function **GetFibonacci(n)** returns the n'th Fibonacci number we can express it using **GetFibonacci(n) = GetFibonacci(n-1) + GetFibonacci(n-2).**

However, this will never end and in a few seconds a StackOverflow Exception is thrown. In order for the recursion to stop it has to have a "**bottom**". The bottom of the recursion is **GetFibonacci(2)** should return 1 and **GetFibonacci(1)** should return 1.

Input Format:

- On the only line in the input the user should enter the wanted Fibonacci number.

Output Format:

- The output should be the n'th Fibonacci number counting from 1.

Constraints:

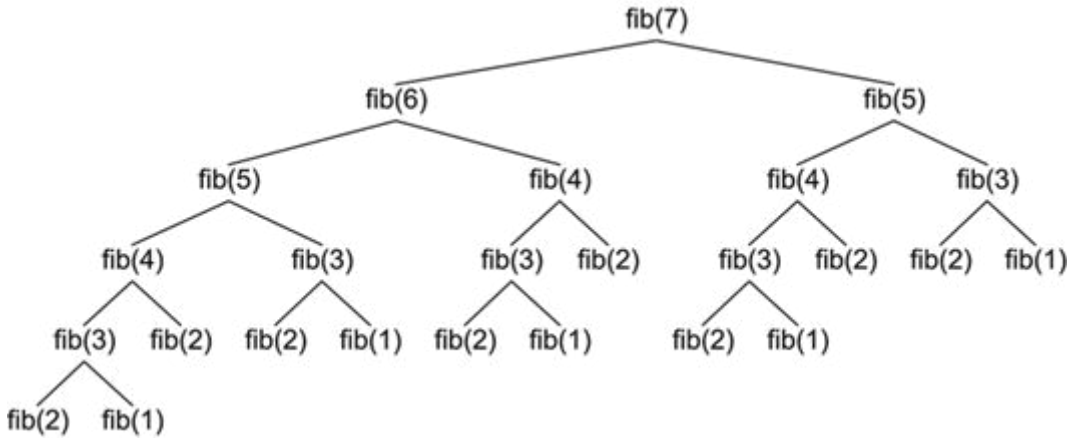
- 1 ≤ N ≤ 50

Examples

Input	Output
5	5
10	55
21	10946

For the Nth Fibonacci number, we calculate the N-1th and the N-2th number, but for the calculation of N-1th number we calculate the N-1-1th(N-2th) and the N-1-2th number, so we have a lot of repeated calculations.

SoftUni
© SoftUni – https://softuni.org. Copyrighted document. Unauthorized copy, reproduction or use is not permitted.
Follow us:



If you want to figure out how to skip those unnecessary calculations, you can search for a technique called memoization.

4. Longest Increasing Subsequence (LIS)

Read a **list of integers** and find the **longest increasing subsequence** (LIS). If several such exist, print the **leftmost**.

Examples

Input	Output
1	1
7 **3 5** 8 -1 **0 6 7**	3 5 6 7
1 2 5 **3 5** 2 4 1	1 2 3 5
0 10 20 30 30 40 **1** 50 **2 3 4 5 6**	0 1 2 3 4 5 6
11 12 13 **3** 14 **4** 15 **5 6 7 8** 7 **16** 9 8	3 4 5 6 7 8 16
3 14 **5** 12 15 **7 8 9 11** 10 1	3 5 7 8 9 11

Hints

- Assume we have **n** numbers in an array **nums[0…n-1]**.
- Let **len[p]** holds the length of the longest increasing subsequence (LIS) ending at position **p**.
- In a for loop, we shall calculate **len[p]** for **p** = **0** … **n-1** as follows:
  - Let **left** is the leftmost position on the left of **p** (**left** < **p**), such that **len[left]** is the largest possible.
  - Then, **len[p]** = **1** + **len[left]**. If **left** does not exist, **len[p]** = **1**.
  - Also, save **prev[p]** = **left** (we hold if **prev[]** the previous position, used to obtain the best length for position **p**).
- Once the values for **len[0…n-1]** are calculated, restore the LIS starting from position **p** such that **len[p]** is maximal and go back and back through **p** = **prev[p]**.
- The table below illustrates these computations:

index	0	1	2	3	4	5	6	7	8	9	10
nums[]	**3**	**14**	**5**	**12**	**15**	**7**	**8**	**9**	**11**	**10**	**1**
len[]	1	2	2	3	4	3	4	5	6	6	1
prev[]	-1	0	0	2	3	2	5	6	7	7	-1
LIS	{3}	{3,14}	{3,5}	{3,5,12}	{3,5,12,15}	{3,5,7}	{3,5,7,8}	{3,5,7,8,9}	{3,5,7,8,9,11}	{3,5,7,8,9,10}	{1}

SoftUni
© SoftUni – https://softuni.org. Copyrighted document. Unauthorized copy, reproduction or use is not permitted.
Follow us:

© SoftUni – https://softuni.org. Copyrighted document. Unauthorized copy, reproduction or use is not permitted.