Lab: Methods

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

You can check your solutions in Judge.

I. Declaring and Invoking Methods

1. Sign of Integer

Create a method that prints the sign of an integer number.

Example

Input	Output
2	The number 2 is positive.
-5	The number -5 is negative.
0	The number 0 is zero.

2. Grades

Write a method that **receives a grade** between **2.00** and **6.00** and **prints the corresponding grade in words:**

- 2.00 – 2.99 - "Fail"
- 3.00 – 3.49 - "Poor"
- 3.50 – 4.49 - "Good"
- 4.50 – 5.49 - "Very good"
- 5.50 – 6.00 - "Excellent"

Examples

Input	Output
3.33	Poor
4.50	Very good
2.99	Fail

Hint

Read the grade from the console and pass it to a method:

```
import java.util.Scanner;

public class Grades {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        double grade = Double.parseDouble(sc.nextLine());
        printInWords(grade);
    }
```

Then create the method and make the `if` statements for each case:

```
private static void printInWords(double grade){
    if(grade >= 2.00 && grade <= 2.99){
        System.out.println("Fail");
    }

    //TODO: make the rest
}
```

3. Printing Triangle

Create a method for printing triangles as shown below:

Examples

Input	Output
3	1 1 2 1 2 3 1 2 1
4	1 1 2 1 2 3 1 2 3 4 1 2 3 1 2 1

Hints

After you read the input, create a method **for printing a single line** from a **given start** to a **given end**. Choose a **meaningful name** for it, describing its purpose:

```
private static void printLine(int start, int end){
    for (int i = start; i <= end; i++) {
        System.out.print(i + " ");
    }
    System.out.println();
}
```

You will need two loops. In the first loop you can print the first half of the triangle without the middle line:

```
for (int i = 0; i < n; i++) {
    printLine(1, i);
}
```

Next, print the middle line:

```
printLine(1, i);
```

Lastly, print the rest of the triangle:

```
for (int i = n - 1; i >= 1; i--) {
    printLine(1, i);
}
```

4. Calculations

Write a program that receives a string on the first line **("add", "multiply", "subtract", "divide")** and on the next two lines receives **two** numbers. Create four methods (for each calculation) and invoke the right one depending on the command. The method should also print the result (needs to be void).

Example

Input	Output
subtract 5 4	1
divide 8 4	2

Hints

Read the command on the first line and the two numbers, and then make an if/switch statement for each type of calculation:

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

    String command = sc.nextLine();
    int a = Integer.parseInt(sc.nextLine());
    int b = Integer.parseInt(sc.nextLine());

    switch (command){
        case "add":
            add(a,b);
            break;
        case "divide":
            divide(a,b);
            break;
        // TODO: check for the rest command
    }
}
```

Then create the four methods and print the result:

```
private static void add(int a, int b){
    System.out.println(a + b);
}

private  static void divide(int a, int b){
    System.out.println(a / b);
}

// TODO: create the rest of the methods
```

5. Orders

Write a method that calculates the total price of an order and prints it on the console. The method should receive **one of the following products**: coffee, water, coke, snacks; and a **quantity** of the product. The prices for a single piece of each product are:

- coffee – 1.50
- water – 1.00
- coke – 1.40
- snacks – 2.00

Print the result rounded to the **second** decimal place.

Example

Input	Output
water 5	5.00
coffee 2	3.00

Hint

- Read the product and the quantity.
- Create a method the pass the two variables in.
- Print the result in the method.

II. Returning Values and Overloading

6. Calculate Rectangle Area

Create a method that calculates and **returns** the area of a rectangle by given width and length.

Examples

Input	Output
3 4	12

Hints

After reading the input, create a method, but this time **instead** of typing **"static void"** before its name, type **"static double"** as this will make it to **return a value of type double**:

```
private static double getRectangleArea(double width, double height) {
    return width * height;
}
```

Invoke the method in the main and **save the return value in a new variable**:

```
double width = Double.parseDouble(sc.nextLine());
double height = Double.parseDouble(sc.nextLine());
double area = getRectangleArea(width, height);
System.out.println(area);
```

7. Repeat String

Write a method that receives a string and a repeat count **n** (integer). The method should return a new string (the old one repeated n times).

Example

Input	Output
abc 3	abcabcabc
String 2	StringString

Hints

Firstly read the string and the repeat count n. Then create the method and pass it the variables:

```
private static String repeatString(String str, int count) {
    String result = "";

    for (int i = 0; i < count; i++) {
        // TODO: append the string to the result
    }

    return result;
}
```

8. Math Power

Create a method that calculates and returns the value of a number raised to a given power.

Examples

Input	Output
2	256

8	
5.5 3	166.375

Hints

Create a method which will have two parameters - the number and the power, and will return a result of type **double**:

```
private static double mathPower(double number, int power) {
    double result = 1;

    //TODO: Calculate result (use a loop or Math.pow())
    return result;
}
```

Invoke the method and use **`DecimalFormat`** to print the result. For the curious - you can read more here.

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

    double number = Double.parseDouble(sc.nextLine());
    int power = Integer.parseInt(sc.nextLine());

    System.out.println(new DecimalFormat( pattern: "0.####").format(mathPower(number, power)));
}
```

9. Greater of Two Values

You are given two values of the same type as input. The values can be of type **`int`**, **`char`** of **`String`**. Create a method **`getMax()`** that returns the greater of the two values:

Examples

Input	Output
int 2 16	16
char a z	z
string Ivan Todor	Todor

Hints

1. For this method you need to create three methods with the same name and different signatures.
2. Create a method which will compare integers:

```
static int getMax(int firstNum, int secondNum) {
    if (firstNum > secondNum) {
        return firstNum;
    }

    return secondNum;
}
```

3. Create a second method with the same name which will compare characters. Follow the logic of the previous method:

```
static char getMax(char first, char second) {
    //TODO: Implement the method
}
```

4. Lastly you need to create a method to compare strings. This is a bit different as strings don't allow to be compared with the operators "**>**" and "**<**":

```
static String getMax(String first, String second) {
    if (first.compareTo(second) >= 0) {
        //TODO: Return value
    }
    //TODO: Return value
}
```

   You need to use the method "**`compareTo()`**", which returns an integer value (greater than zero if the compared object is greater, less than zero if the compared object is lesser and zero if the two objects are equal).

5. The last step is to read the input. Use appropriate variables and call the **`getMax()`** from your **`main()`**.

10. Multiply Evens by Odds

Create a program that reads an **integer** number and **multiplies the sum of all its even digits** by **the sum of all its odd digits**:

Examples

Input	Output	Comments
12345	54	12345 has **2 even digits** - 2 and 4. Even digits have **sum of 6.** Also it has **3 odd digits** - 1, 3 and 5. Odd digits have **sum of 9.** **Multiply 6 by 9** and you get **54.**
-12345	54	

Hints

1. Create a method with a **name describing its purpose** (like **`getMultipleOfEvensAndOdds`**). The method should have a **single integer parameter** and an **integer return value**. Also, the method will call two other methods:

```
private static int getMultipleOfEvensAndOdds(int n) {
    int evensSum = getSumOfEvenDigits(n);
    int odsSum = getSumOfOddDigits(n);

    return evensSum * odsSum;
}
```

2. Create two other methods each of which will sum either even or odd digits.
3. Implement the logic for summing even digits:

```
private static int getEvenSum(int n){
    int evensSum = 0;

    // TODO check the digits of the number and if they are even, sum them

    return evensSum;
}
```

4. Do the same for the method that will sum odd digits.
5. As you test your solution you may notice that it doesn't work for negative numbers. Following the program execution line by line, find and fix the bug (**hint: you can use `Math.abs()`**).

11. Math operations

Write a method that receives **two numbers** and **an operator**, calculates the result and returns it. You will be given three lines of input. The first will be the first number, the second one will be the operator and the last one will be the second number. The possible operators are: **`/ * + -`**

Print the result rounded up to the **second** decimal point.

Example

Input	Output
5 * 5	25
4 + 8	12

Hint

Read the inputs and create a method that returns a double (the result of the operation):

```
private static double calculate(int a, String operator, int b){
    double result = 0.0;

    switch (operator){
        // TODO: check for all possible operands and
        // TODO: calculate the result
    }

    return result;
}
```

Exercises: Methods

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

You can check your solutions in Judge.

1. Smallest of Three Numbers

Write a method to print the smallest of three integer numbers. Use appropriate name for the method.

Examples

Input	Output
2 5 3	2
600 342 123	123
25 -21 4	-21

2. Vowels Count

Write a method that receives a single string and prints the count of the vowels. Use appropriate name for the method.

Examples

Input	Output
SoftUni	3
Cats	1
JS	0

3. Characters in Range

Write a method that receives two characters and prints on a single line all the characters in between them according to ASCII.

Examples

Input	Output
a d	b c
# :	$ % & ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9

C #	$ % & ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; < = > ? @ A B

4. Password Validator

Write a program that checks if a given password is valid. Password rules are:

- 6 – 10 characters **(inclusive)**;
- Consists only of **letters** and **digits**;
- Have at least **2** digits.

If a password is valid, print **"Password is valid"**. If it is not valid, for every unfulfilled rule print a message:

- **"Password must be between 6 and 10 characters"**;
- **"Password must consist only of letters and digits"**;
- **"Password must have at least 2 digits"**.

Examples

Input	Output
logIn	`Password must be between 6 and 10 characters` `Password must have at least 2 digits`
MyPass123	`Password is valid`
Pass	`Password must consist only of letters and digits` `Password must have at least 2 digits`

Hints

Write a method for each rule.

5. Add and Subtract

You will receive 3 **integers.** Write a method **`sum`** to get the sum of the first two integers and **`subtract`** method that subtracts the third integer from the result from the `sum` method.

Examples

Input	Output
23 6 10	19
1 17 30	-12
42 58 100	0

6. Middle Characters

You will receive a single string. Write a method that prints the middle character. If the length of the string is even, there are two middle characters.

Examples

Input	Output
aString	r
someText	eT
3245	24

7. NxN Matrix

Write a method that receives a single integer **n** and prints **nxn** matrix with that number.

Examples

Input	Output
3	3 3 3 3 3 3 3 3 3
7	7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
2	2 2 2 2

8. Factorial Division

Read two integer numbers. Calculate factorial of each number. Divide the first result by the second and print the division formatted to the second decimal point.

Examples

Input	Output
5 2	60.00

Input	Output
6 2	360.00

9. Palindrome Integers

A **palindrome** is a number which reads the same backward as forward, such as 323 or 1001. Write a program which reads a positive integer numbers until you receive "**END**". For each number print whether the number is palindrome or not.

Examples

Input	Output
123 323 421 121 END	false true false true

Input	Output
32 2 232 1010 END	false true true false

10. Top Number

Read an **integer** n from the console. Find all top numbers in the range **[1 … n]** and print them. A top number holds the following properties:

- Its **sum of digits is divisible by 8**, e.g. 8, 16, 88.
- Holds at least **one odd digit**, e.g. 232, 707, 87578.

Examples

Input	Output
50	17 35

Input	Output
100	17 35 53 71 79 97

11. *Array Manipulator

Trifon has finally become a junior developer and has received his first task. It's about manipulating an array of integers. He is not quite happy about it, since he hates manipulating arrays. They are going to pay him a lot of money, though, and he is willing to give somebody half of it if to help him do his job. You, on the other hand, love arrays (and money) so you decide to try your luck.

The array may be manipulated by one of the following commands:

- **exchange {index}** – splits the array **after** the given index and exchanges the places of the two resulting subarrays. E.g. [1, 2, 3, 4, 5] -> **exchange 2** -> result: **[4, 5, 1, 2, 3]**
  - If the index is outside the boundaries of the array, print "**Invalid index**".
- **max even/odd**– returns the **INDEX** of the max even/odd element -> [1, 4, 8, 2, 3] -> **max odd** -> print **4**
- **min even/odd** – returns the **INDEX** of the min even/odd element -> [1, 4, 8, 2, 3] -> **min even** > print **3**
  - If there are two or more equal **min/max** elements, return the index of the **rightmost** one.
  - If a **min/max even/odd** element **cannot** be found, print **"No matches"**.
- **first {count} even/odd**– returns the first {count} elements -> [1, 8, 2, 3] -> **first 2 even** -> print **[8, 2]**
- **last {count} even/odd** – returns the last {count} elements -> [1, 8, 2, 3] -> **last 2 odd** -> print **[1, 3]**
  - If the count is greater than the array length, print "**Invalid count**".
  - If there are **not enough** elements to satisfy the count, print as many as you can. If there are **zero even/odd** elements, print an empty array **"[]"**.
- **end** – stop taking input and print the final state of the array.

Input

- The input data should be read from the console.

- On the first line, the initial array is received as a line of integers, separated by a single space.
- On the next lines, until the command "**end**" is received, you will receive the array manipulation commands.
- The input data will always be valid and in the format described. There is no need to check it explicitly.

Output

- The output should be printed on the console.
- On a separate line, print the output of the corresponding command
- On the last line, print the final array in **square brackets** with its elements separated by a comma and a space
- See the examples below to get a better understanding of your task

Constraints

- The **number of input lines** will be in the range [2 … 50].
- The **array elements** will be integers in the range [0 … 1000].
- The **number of elements** will be in the range [1 ... 50]
- The **split index** will be an integer in the range [-2^{31} … $2^{31} - 1$]
- **first/last count** will be an integer in the range [1 … $2^{31} - 1$]
- There will **not** be redundant whitespace anywhere in the input
- Allowed working time for your program: 0.1 seconds. Allowed memory: 16 MB.

Examples

Input	Output
1 3 5 7 9 exchange 1 max odd min even first 2 odd last 2 even exchange 3 end	2 No matches [5, 7] [] [3, 5, 7, 9, 1]
1 10 100 1000 max even first 5 even exchange 10 min odd exchange 0 max even min even end	3 Invalid count Invalid index 0 2 0 [10, 100, 1000, 1]
1 10 100 1000 exchange 3 first 2 odd last 4 odd end	[1] [1] [1, 10, 100, 1000]

More Exercises: Methods

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

You can check your solutions in Judge.

1. Data Types

Write a program that, depending on the first line of the input, reads an **int**, **double** or **string**.

- If the data type is **int**, multiply the number by 2.
- If the data type is **real**, multiply the number by 1.5 and format it to the second decimal point.
- If the data type is **string**, surround the input with "$".

Print the result on the console.

Examples

Input	Output
int 5	10
real 2	3.00
string hello	$hello$

Hint

Try to solve the problem using only one method with different overloads.

2. Center Point

You are given the coordinates of two points on a Cartesian coordinate system - X1, Y1, X2 and Y2. **Create a method** that prints the point that is closest to the center of the coordinate system (0, 0) in the format (X, Y). If the points are on a same distance from the center, print only the first one.

Examples

Input	Output
2 4 -1 2	(-1, 2)

3. Longer Line

You are given the coordinates of four points in the 2D plane. The first and the second pair of points form two different lines. Print the longer line in format "(X1, Y1)(X2, Y2)" starting with the point that is closer to the center of the coordinate system (0, 0) (You can reuse the method that you wrote for the previous problem). If the lines are of equal length, print only the first one.

Examples

Input	Output
2 4 -1 2 -5 -5 4 -3	(4, -3)(-5, -5)

4. Tribonacci Sequence

In the **"Tribonacci" sequence**, every number is formed by the **sum of the previous 3**.

You are given a number **num**. Write a program that prints **num** numbers from the Tribonacci sequence, each on a new line, starting from 1. The input comes as a parameter named **num**. The value **num** will always be positive integer.

Examples

Input	Output
4	1 1 2 4

Input	Output
8	1 1 2 4 7 13 24 44

5. Multiplication Sign

You are given a number **num1**, **num2** and **num3**. Write a program that finds if **num1 * num2 * num3** (the product) is **negative**, **positive or zero**. Try to do this **WITHOUT** multiplying the 3 numbers.

Examples

Input	Output
2 3 -1	negative

Input	Output
2 3 1	positive