More Exercises: Methods

Problems for exercises and homework for the ["Programming 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 a **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 at the same distance from the center, print only the first one.

Examples

Input	Output
2 4 -1 2	(-1, 2)
4 -2 5 1	(4, -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 the 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 6 -2 -1 2 4 7 3	`(-2, -1)(4, 6)`

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 a 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`

6. 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 will 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} ... $2^{31} – 1$]**.
- **The 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`	`3`

`max even`
`first 5 even`
`exchange 10`
`min odd`
`exchange 0`
`max even`
`min even`
`end`
	`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]`