Exercises: Methods

Problems for exercise and homework for the "Programming Fundamentals: Arrays and Lists" course from the official "Applied Programmer" curriculum.

You can check your solutions in https://judge.softuni.bg/Contests/2910.

1. Smallest of Three Numbers

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

Examples

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

2. Vowels Count

Write a method that receives a **single string** and prints the **count of the vowels**. Use an 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 between them** according to **ASCII** table.

Examples

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

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. 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

6. Factorial Division

Read **two integer numbers**. Calculate [factorial](https://softuni.org) 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

7. 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

8. Top Number

A **top number** is an integer that 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.

Write a program to print **all top numbers** in the range **[1…n]**.

Examples

Input	Output
50	17 35

Input	Output
100	17 35 53 71 79 97

9. *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 sub-arrays. 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]

Input	Output
1 10 100 1000 max even first 5 even exchange 10 min odd exchange 0 max even	3 Invalid count Invalid index 0 2 0 [10, 100, 1000, 1]

min even end	
Input	**Output**
1 10 100 1000 exchange 3 first 2 odd last 4 odd end	[1] [1] [1, 10, 100, 1000]