Exercise: Methods

Problems for exercise and homework for the ["C# Fundamentals" course @ SoftUni](#)

You can check your solutions in [Judge](#)

1. Smallest of Three Numbers

Create a method that **prints out the smallest of three integer numbers**.

Examples

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

2. Vowels Count

Create a method that receives a **single string** and **prints out the number of vowels** contained in it.

Examples

Input	Output
SoftUni	3
Cats	1
JS	0

3. Characters in Range

Create a method that receives **two characters** and prints all the **characters between them according to ASCII** (on a single line).

NOTE: If the second letter's ASCII value is less than that of the first one, then the two initial letters should be swapped.

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

Create a program that checks if a given password is **valid**.

The password validation **rules** are:

- It should contain **6 – 10 characters (inclusive)**
- It should contain **only letters and digits**
- It should contain **at least 2 digits**

If it is **not valid**, for any unfulfilled rule **print the corresponding 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
Pa$s$s	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. Create a method that returns the sum of the first two integers and another method that subtracts the third integer from the result of 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. Create a method that **prints the character found at its middle**. 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

Create a method that receives a single integer **n** and prints an **NxN** matrix using this number as a filler.

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 integers**. Calculate the factorial of each number. **Divide the first result by the second** and print the result of the division **formatted to the second decimal point**.

Examples

Input	Output

`5`
`2`
	`60.00`

Input	Output

`6`
`2`
	`360.00`

9. Palindrome Integers

Create a program that reads positive integers **until you receive** the **"END"** command. For each number, **print whether the number is a palindrome or not**. A palindrome is a number that reads the same backward as forward, such as 323 or 1001.

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

A **top number** is an integer that holds the following properties:

- Its sum of digits is divisible by 8, e.g. 8, 17, 88
- Holds at least one odd digit, e.g. 232, 707, 87578
- Some examples of top numbers are: 17, 161, 251, 4310, 123200

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

You will receive a single integer from the console, representing the end value.

Examples

Input	Output
50	17 35

Input	Output
100	17 35 53 71 79 97

11. *Array Manipulator

Peter 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 they 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].
- **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]`

Input	**Output**

`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]`

Input	**Output**

`1 10 100 1000`
`exchange 3`
	`[1]`
	`[1]`

`first 2 odd`
`last 4 odd`
`end`
	`[1, 10, 100, 1000]`