# Lab: Data Types, Variables and Methods

Problems for exercises and homework for the [“Programming Fundamentals Extended” course @ SoftUni](https://softuni.bg/courses/programming-fundamentals).

## Rotate Array of Strings

Write a program to read **an array of strings**, **rotate** it to the right and **print** its rotated elements.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| a b c d e | e a b c d |
| soft uni hi | hi soft uni |
| i r a b | b i r a |

### Hints:

* You can store the rotated array in a **second array** alongside the first one

## Odd Numbers at Odd Positions

Write a program to read an array of integers and find how many **odd numbers** at **odd positions** the array holds. If there are no numbers which match this criteria, **do not print anything**

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Explanation** |
| 2 **3** 5 2 7 **9** -1 **-7** | Index 1 -> 3  Index 5 -> 9  Index 7 -> -7 | Indexes: 0 **1** 2 3 4 **5** 6 **7**  Numbers: 2 **3** 5 2 7 **9** **-1** -7  Odd positions with odd numbers: 1, 5 and 7 |
| 2 **3** 55 2 4 **1** | Index 1 -> 3  Index 5 -> 1 | Indexes: 0 **1** 2 3 4 **5**  Numbers: 2 **3** 55 2 4 **1**  Odd positions with odd numbers: 1 and 5 |
| 5 0 1 2 | *(no output)* | Indexes: 0 1 2 3  Numbers: 5 0 1 2  Odd positions with odd numbers: **none** |

### Hints

* Positions are counted **from** **0** from left to right, so if for example the second element (**index 1**) is **odd**, then we **should** count it, and so on…
* Do **NOT** count odd numbers which are at **even** positions (0, 2, 4, etc…)

## Array Contains Element

Read a **list of integers** on the first line of the console and an integer **N** from the second line of the console and print whether the element is **contained** in the array. If it is, print “yes”, otherwise print “no”.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 1 2 3 4 5  5 | yes |
| 8 7 7 9 6 2 2  11 | no |
| 99 7 8 6 2314 2  2314 | yes |

### Hints

* Read a text line from the console, split it by space, parse the obtained items as integers and convert them to array of integers.
* Scan through the whole array, element by element, until you either find the element, or reach the end of the array.
* Keep the result of the operation in a Boolean variable such as “isFound”.
* Finally, if the element is found (checking by the variable), print “yes” or “no”.

## Sort Numbers

Read a **list of decimal numbers** and **sort** them in increasing order. Print the output as shown in the examples below.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 8 2 7 3 | 2 <= 3 <= 7 <= 8 |
| 2 4 -9 | -9 <= 2 <= 4 |

### Hints

* Use the built-in method List<T>.Sort().

## Largest N Elements

Read a **list of integers** on the first line of the console. On the next line, you will receive an **integer N**. After that, find and **print** the **largest N** **elements** the array, sorted in **descending order**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 5 3 4 1 2  3 | 5 4 3 |
| 11 872 673 1 2  2 | 872 673 |
| 11 52 43 12 1 6  4 | 52 43 12 11 |

### Hints

* A possible solution to this problem is:
  + Sort the array in **descending order**, using a sorting algorithm such as **Insertion Sort**,
  + Extract the first **N elements** from it