# Lab: Arrays, Multidimensional Arrays and Associative Arrays

You can check your solutions here: <https://judge.softuni.bg/Contests/1249/Advanced-Syntax-and-Operations-Lab>

# Part: Arrays

## Print Month

Enter a **month number** [1…12] and print the **month name** (in English) or “**Invalid Month!**”. Use an **array of strings**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2 | February |
| 8 | August |
| 12 | December |
| 14 | Invalid Month! |

### Hints

* Use an **array of strings** holding the day names: {"January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"}.
* Print the element at index (**month-1**) when it is in the range [1…12] or “**Invalid Month!**” otherwise.

## Reverse an Array of Integers

Write a program to read **an array of integers**, **reverse** it and **print** its elements. The input consists of a **number** n (the number of elements) + n integers, each as a separate line. Print the output on a single line (space separated).

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| **3**  10  20  30 | 30 20 10 |
| **4**  -1  20  99  5 | 5 99 20 -1 |

### Hints

* First, read the number n.
* Allocate an array of n integers.
* Read the integers in a for-loop.
* Instead of reversing the array, you can just pass through the elements from the last (**n-1**) to the first (**0**) with a reverse for-loop.

## Triple Sum

Write a program to read **an array of integers** and find all triples of elements **a**, **b** and **c**, such that **a** + **b** == **c** (where **a** stays to the left from **b**). Print “**No**” if no such triples exist.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 1 1 1 1 | No |
| 4 2 8 6 | 4 + 2 == 6  2 + 6 == 8 |
| 2 7 5 0 | 2 + 5 == 7  2 + 0 == 2  7 + 0 == 7  5 + 0 == 5 |
| 3 1 5 6 1 2 | 3 + 2 == 5  1 + 5 == 6  1 + 1 == 2  1 + 2 == 3  5 + 1 == 6  1 + 2 == 3 |

### Hints:

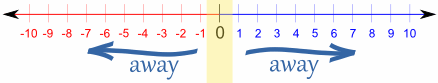
* Read the input numbers in array arr[].
* Use nested loops to generate all pairs {a, b}, such that 0 ≤ a < b < n.
* Check whether arr[] contains the sum arr[a] + arr[b].

## Rounding Numbers Away from Zero

Write a program to read **an array of real numbers** (space separated values), **round** them to the nearest integer in “**away from 0**” style and **print** the output as in the examples below.

Rounding in “[away from zero](https://www.mathsisfun.com/numbers/rounding-methods.html)” style means:

* To round to the nearest integer, e.g. 2.9 🡪 3; -1.75 🡪 -2
* In case the number is exactly in the middle of two integers (midpoint value), round it to the next integer which is away from the 0:



### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 0.9 1.5 2.4 2.5 3.14 | 0.9 => 1  1.5 => 2  2.4 => 2  2.5 => 3  3.14 => 3 |
| -5.01 -1.599 -2.5 -1.50 0 | -5.01 => -5  -1.599 => -2  -2.5 => -3  -1.50 => -2  0 => 0 |

### Hints:

* **Approach I**: Take the **absolute value** of each input number, add **0.5** and **truncate** the integral part. If the original number is negative, make the result also negative.
* **Approach II**: **Search in Internet** for **“rounding away from zero” + PHP**. In this case it’s **“PHP\_ROUND\_HALF\_UP”.**  You should find a built-in [PHP method for rounding](http://php.net/manual/en/function.round.php) in many styles. Choose the “away from zero” rounding.
  + For example **echo round(9.5 , 0 ,PHP\_ROUND\_HALF\_UP)** Will print **10**.

Here first parameter in this case **9.5** is the number we want to round.

Second parameter in this case **0** is the precision we want (0 means 0 digits after decimal point).

Third parameter in this case **PHP\_ROUND\_HALF\_UP** is the method we want to use for rounding.

## Sum Arrays

Write a program that reads two **arrays of integers** and sums them. When the arrays are not of the same size, duplicate the smaller array a few times.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 1 2 3 4  2 3 4 5 | 3 5 7 9 | 1 2 3 4 +  2 3 4 5 =  3 5 7 9 |
| 1 2 3 4 5  2 3 | 3 5 5 7 7 | 1 2 3 4 5 +  2 3 2 3 2 =  3 5 5 7 7 |
| 5 4 3  2 3 1 4 | 7 7 4 9 | 5 4 3 5 +  2 3 1 4 +  7 7 4 9 |

### Hints

* Assume the first array arr1 has len1 elements and the second arr2 has len2 elements.
* The result array will have max(len1, len2) elements.
* We sum array elements one by one (from the first to the last). To enable **rotating** (take the first element as next after the last), we use the position % length indexing: arr1[i % len1] and arr2[i % len2].

# Part: Multidimensional Arrays

## Fill the Matrix

Filling a matrix in the regular way (**top to bottom** and **left to right**) is boring. Write code that **fill** a **matrix** of size **N x** N in **two** different **patterns.** Both patterns are described below:

| **Pattern A** | **Pattern B** |
| --- | --- |
|  |  |

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3, A | 1 4 7  2 5 8  3 6 9 |
| 3, B | 1 6 7  2 5 8  3 4 9 |

### Hints

* Make a different method for each pattern
* Make a method for printing the matrix

## Find Min and Max Element

Write program that **read a matrix** from console and print min and max element:

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3, 6  7, 1, 3, 3, 2, 1 1, 3, 9, 8, 5, 6 4, 6, 7, 9, 1, 1 | Min: 1  Max: 9 |

## Diagonal Difference

Write a program that finds the **difference between the sums of the square matrix diagonals** (absolute value).



### Input

* The **first line** holds a number n – the size of matrix.
* The next n **lines** hold the **values for every row** – n numbers separated by a space.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 3  11 2 4  4 5 6  10 8 -12 | 15 | **Primary diagonal:** sum = 11 + 5 + (-12) = 4  **Secondary diagonal:** sum = 4 + 5 + 10 = 19  **Difference:** |4 - 19| = 15 |

# Part: Associative Arrays

## Count Real Numbers

Read a **list of real numbers** and **print them in ascending order** along with their **number of occurrences**.

### Examples

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |
| 8 2.5 2.5 8 2.5 | 2.5 -> 3  8 -> 2 | 1.5 5 1.5 3 | 1.5 -> 2  3 -> 1  5 -> 1 | -2 0.33 0.33 2 | -2 -> 1  0.33 -> 2  2 -> 1 |

### Hints

* Use sort array in ascending order.
* Pass through each input number num and increase counts[num] (when num exists in the dictionary) or assign counts[num] = 1 (when num does not exist in the dictionary).

## Odd Occurrences

Write a program that extracts from a given sequence of words all elements that present in it **odd number of times** (case-insensitive).

* Words are given in a single line, space separated.
* Print the result elements in lowercase, in their order of appearance.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Java C# PHP PHP JAVA C java | java, c#, c |
| 3 5 5 hi pi HO Hi 5 ho 3 hi pi | 5, hi |
| a a A SQL xx a xx a A a XX c | a, sql, xx, c |

### Hints

* Use a **associative array** (string 🡪 int) to count the occurrences of each word (just like in the previous problem).
* Pass through all **key-value pairs** in the dictionary and append to the results array all **keys** that have **odd value**.
* Print the results list.

## Letter Repetition

You will be given a **single string**, containing **random ASCII character**. Your task is to **print every character**, and how **many times** it has been used in the **string**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| aaabbaaabbbccc | a -> 6  b -> 5  c -> 3 |
| The quick brown fox jumps over the lazy dog | T -> 1  h -> 2  e -> 3  -> 8  q -> 1  u -> 2  i -> 1  c -> 1  k -> 1  b -> 1  r -> 2  o -> 4  w -> 1  n -> 1  f -> 1  x -> 1  j -> 1  m -> 1  p -> 1  s -> 1  v -> 1  t -> 1  l -> 1  a -> 1  z -> 1  y -> 1  d -> 1  g -> 1 |

## Sum by Town

### Read Towns and Incomes (Like Shown Below) and Print an Array Holding the Total Income for Each Town (See Below). Print the Towns in Their Natural Order As Object Properties.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Sofia, 20, Varna, 10, Sofia, 5 | Sofia => 25  Varna => 10 |
| Plovdiv, 40, Pernik, 20, Vidin, 8, Sliven, 44, Plovdiv, 1, Vidin, 7, Chirpan, 0 | Plovdiv => 41  Pernik => 20  Vidin => 15  Sliven => 44  Chirpan => 0 |

## Periodic Table

You are given an n number of chemical elements. You need to keep track of all elements and at the end print all **unique ones**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Ce, O, Mo, O, Ce, Ee, Mo | Ee |
| Ge, Ch, O, Ne, Nb, Mo, Tc, O, Ne | Ge Ch Nb Mo Tc |