# Homework: PHP Arrays Classical

This document defines the homework assignments from the ["PHP Fundamentals" Course @ Software University](https://softuni.bg/courses/php-basics/). Please submit as homework a single zip / rar / 7z archive holding the solutions (source code) of all below described problems.

## Largest Common End

Read **two** **arrays** **of** **words** and find the length of the **largest common end** (left or right).

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| **hi php java** csharp sql html css js  **hi php java** js softuni nakov java learn | 3 | The largest common end is at the left: **hi php java** |
| hi php java xml csharp **sql html css js**  nakov java **sql html css js** | 4 | The largest common end is at the right: **sql html css js** |
| I love programming  Learn Java or C# | 0 | No common words at the left and right |

### Hints

* Scan the arrays from left to right until the end of the shorter is reached and count the equal elements.
* Scan the arrays form right to left until the start of the shorter is reached.
* Keep the start position and the length of the longest equal start / end.

## Rotate and Sum

To “**rotate** an array on the right” means to move its last element first: {1, 2, 3} 🡪 {3, 1, 2}.

Write a program to read an array of **n** **integers** (space separated on a single line) and an integer **k**, rotate the array right **k** **times** and sum the obtained arrays after each rotation as shown below.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 3 2 4 -1  2 | 3 2 5 6 | rotated1[] = -1 3 2 4  rotated2[] = 4 -1 3 2  sum[] = 3 2 5 6 |
| 1 2 3  1 | 3 1 2 | rotated1[] = 3 1 2  sum[] = 3 1 2 |
| 1 2 3 4 5  3 | 12 10 8 6 9 | rotated1[] = 5 1 2 3 4  rotated2[] = 4 5 1 2 3  rotated3[] = 3 4 5 1 2  sum[] = 12 10 8 6 9 |

### Hints

* After r rotations the element at position i goes to position (i + r) % n.
* The sum[] array can be calculated by two nested loops: for r = 1 … k; for i = 0 … n-1.

## Max Sequence of Equal Elements

Write a program that finds the **longest sequence of equal elements** in an array of integers. If several longest sequences exist, print the leftmost one.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2 1 1 2 3 3 **2 2 2** 1 | 2 2 2 |
| **1 1 1** 2 3 1 3 3 | 1 1 1 |
| **4 4 4 4** | 4 4 4 4 |
| 0 **1 1** 5 2 2 6 3 3 | 1 1 |

### Hints

* Start with the sequence that consists of the first element: start=0, len=1.
* Scan the elements from left to right, starting at the second element: pos=1…n-1.
  + At each step compare the current element with the element on the left.
    - Same value 🡪 you have found a sequence longer by one 🡪 len++.
    - Different value 🡪 start a new sequence from the current element: start=pos, len=1.
  + After each step remember the sequence it is found to be longest at the moment: bestStart=start, bestLen=len.
* Finally, print the longest sequence by using bestStart and bestLen.

## \*Max Sequence of Increasing Elements

Write a program that finds the **longest increasing subsequence** in an array of integers. The longest increasing subsequence is a **portion of the array** (subsequence) that is strongly **increasing** and has the **longest possible length**. If several such subsequences exist, find the left most of them.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3 **2 3 4** 2 2 4 | 2 3 4 |
| 4 5 **1 2 3 4 5** | 1 2 3 4 5 |
| **3 4 5 6** | 3 4 5 6 |
| **0 1** 1 2 2 3 3 | 0 1 |

### Hints

* Use the same algorithm like in the previous problem (Max Sequence of Equal Elements).

## Most Frequent Number

Write a program that finds the **most frequent number** in a given sequence of numbers.

* Numbers will be in the range [0…65535].
* In case of multiple numbers with the same maximal frequency, print the leftmost of them.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Output** |
| **4** 1 1 **4** 2 3 **4 4** 1 2 **4** 9 3 | 4 | The number **4** is the most frequent (occurs 5 times) |
| **2 2 2 2** 1 **2 2 2** | 2 | The number **2** is the most frequent (occurs 7 times) |
| **7 7 7** 0 2 2 2 0 10 10 10 | 7 | The numbers **2**, **7** and **10** have the same maximal frequence (each occurs 3 times). The leftmost of them is **7**. |

## Index of Letters

Write a program that creates an array containing all letters from the alphabet (**a**-**z**). Read a lowercase word from the console and print the **index of each of its letters in the letters array**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Abcz | a -> 0  b -> 1  c -> 2  z -> 25 |
| softuni | s -> 18  o -> 14  f -> 5  t -> 19  u -> 20  n -> 13  i -> 8 |

## Equal Sums

Write a program that determines if there **exists an element in the array** such that the **sum of the elements on its left** is **equal** to the **sum of the elements on its right**. If there are **no elements to the left / right**, their **sum is considered to be 0**. Print the **index** that satisfies the required condition or **“no”** if there is no such index.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 1 2 3 3 | 2 | At a[2] -> left sum = 3, right sum = 3  a[0] + a[1] = a[3] |
| 1 2 | no | At a[0] -> left sum = 0, right sum = 2  At a[1] -> left sum = 1, right sum = 0  No such index exists |
| 1 | 0 | At a[0] -> left sum = 0, right sum = 0 |
| 1 2 3 | no | No such index exists |
| 10 5 5 99 3 4 2 5 1 1 4 | 3 | At a[3] -> left sum = 20, right sum = 20  a[0] + a[1] + a[2] = a[4] + a[5] + a[6] + a[7] + a[8] + a[9] + a[10] |

## \*\* Longest Increasing Subsequence (LIS)

Read a **list of integers** and find the **longest increasing subsequence** (LIS). If several such exist, print the **leftmost**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| **1** | 1 |
| 7 **3 5** 8 -1 0 **6 7** | 3 5 6 7 |
| **1 2** 5 **3 5** 2 4 1 | 1 2 3 5 |
| **0** 10 20 30 30 40 **1** 50 **2 3 4 5 6** | 0 1 2 3 4 5 6 |
| 11 12 13 **3** 14 **4** 15 **5 6 7 8** 7 **16** 9 8 | 3 4 5 6 7 816 |
| **3** 14 **5** 12 15 **7 8 9 11** 10 1 | 3 5 7 8 9 11 |

### Hints

* Assume we have n numbers in an array nums[0…n-1].
* Let len[p] holds the length of the longest increasing subsequence (LIS) ending at position p.
* In a for loop, we calculate shall len[p] for p = 0 … n-1 as follows:
  + Let left is the leftmost position on the left of p (left < p), such that len[left] is the maximal possible.
  + Then, len[p] = 1 + len[left]. If left does not exist, len[p] = 1.
  + Also save prev[p] = left (we hold if prev[] the previous position, used to obtain the best length for position p).
* Once the values for len[0…n-1] are calculated, restore the LIS starting from position p such that len[p] is maximal and go back and back through p = prev[p].
* The table below illustrates these computations:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| nums[] | **3** | **14** | **5** | **12** | **15** | **7** | **8** | **9** | **11** | **10** | **1** |
| len[] | 1 | 2 | 2 | 3 | 4 | 3 | 4 | 5 | 6 | 6 | 1 |
| prev[] | -1 | 0 | 0 | 2 | 3 | 2 | 5 | 6 | 7 | 7 | -1 |
| LIS | {3} | {3,14} | {3,5} | {3,5,12} | {3,5,12,15} | {3,5,7} | {3,5,7,8} | {3,5,7,8,9} | {3,5,7,8,9,11} | {3,5,7,8,9,10} | {1} |

## \* Array Manipulator

Write a program that **reads an array of integers** from the console and **set of commands** and **executes them over the array**. The commands are as follows:

* **add <index> <element>** – adds element at the specified index (elements right from this position inclusively are shifted to the right).
* **addMany <index> <element 1> <element 2> … <element n>** – adds a set of elements at the specified index.
* **contains <element>** – prints the index of the first occurrence of the specified element (if exists) in the array or **-1** if the element is not found.
* **remove <index>** – removes the element at the specified index.
* **shift <positions>** – **shifts every element** of the array the number of positions **to the** **left** (with rotation).
  + For example, [1, 2, 3, 4, 5] -> shift 2 -> [3, 4, 5, 1, 2]
* **sumPairs** – sums the elements in the array by pairs (first + second, third + fourth, …).
  + For example, [1, 2, 4, 5, 6, 7, 8] -> [3, 9, 13, 8].
* **print** – stop receiving more commands and print the last state of the array.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 1 2 4 5 6 7  add 1 8  contains 1  contains -3  print | 0  -1  [1, 8, 2, 4, 5, 6, 7] |
| 1 2 3 4 5  addMany 5 9 8 7 6 5  contains 15  remove 3  shift 1  print | -1  [2, 3, 5, 9, 8, 7, 6, 5, 1] |
| 2 2 4 2 4  add 1 4  sumPairs  print | [6, 6, 6] |
| 1 2 1 2 1 2 1 2 1 2 1 2  sumPairs  sumPairs  addMany 0 -1 -2 -3  print | [-1, -2, -3, 6, 6, 6] |

## Sum Reversed Numbers

Write a program that reads sequence of numbers, reverses their digits, and prints their sum.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 123 234 12 | 774 | 321 + 432 + 21 = 774 |
| 12 12 34 84 66 12 | 220 | 21 + 21 + 43+ 48 + 66 + 21 = 220 |
| 120 1200 12000 | 63 | 21 + 21 + 21 = 63 |