Challenge 1: Anagrams

Two words are considered anagrams if by rearranging the letters of the first word we get the second word. For instance cinema and iceman are anagrams.

Given a list of word pairs

**Your task is to**

* write a function that determines for each pair if it’s an anagram or not
* for each pair of words your function will print to standard output (stdout) the value 1 if the pair is an anagram or 0 otherwise (one result per line)

Note that your function will receive the following arguments:

* **firstWords**
  + which is an array of strings giving the first word for each of the pairs
* **secondWords**
  + which is an array of strings giving the corresponding second word

**Data constraints**

* the number of word pairs will not exceed 100
* the maximum length of any word in the pairs will not exceed 100 characters
* all words will contain only lowercase English letters (a-z)

**Efficiency constraints**

* your function is expected to print the result in less than 2 seconds

**Example**

|  |  |
| --- | --- |
| **Input** | **Output** |
| firstWords: “cinema”, “host”, “aba”, “train” secondWords: “iceman”, “shot”, “bab”, “rain” | 1 1 0 0 |

**Explanation**

* for the given arguments above we have the following pairs:
  + (cinema, iceman) (host, shot) (aba, bab) (train, rain)
* only the first two pairs are anagrams.

Challenge 2: Braces

Given an array of strings containing three types of braces: round (), square [] and curly {}

**Your task is to**

* write a function that checks whether the braces in each string are correctly matched
* prints 1 to standard output (stdout) if the braces in each string are matched and 0 if they're not (one result per line)

Note that your function will receive the following arguments:

* **expressions**
  + which is an array of strings containing braces

**Data constraints**

* the length of the array will not exceed 100
* the length of any string will not exceed 5000

**Efficiency constraints**

* your function is expected to print the result in less than 2 seconds

**Example**

|  |  |
| --- | --- |
| **Input** | **Output** |
| expressions: [ ")(){}", "[]({})", "([])", "{()[]}", "([)]" ] | 0 1 1 1 0 |

Challenge 3: Hill

Given an array of integer elements

**Your task is to**

* write a function that finds the minimum value X that makes possible the following: generate a new array that is sorted in strictly ascending order by increasing or decreasing each of the elements of the initial array with integer values in the [0, X] range.
  + Example: Having the initial array [5, 4, 3, 2, 8] the minimum value for X is 3. Decrease the first element (5) by 3, decrease the second one (4) by 1, increase the third one (3) by 1, increase the forth one (2) by 3 and do nothing to the last one (8). In the end we obtain the array [2, 3, 4, 5, 8] which is sorted in strictly ascending order.
* print the result X to the standard output (stdout)

Note that your function will receive the following arguments:

* **v**
  + which is the array of integers

**Data constraints**

* numbers are in ascending order when arranged from the smallest to the largest number
* strictly ascending order means that each element is greater than the preceding one (e.g. [1, 2, 2, 3] is NOT strictly ascending order)
* the length of the array will not exceed 5000
* the elements of any of the arrays are integer numbers in the [1, 231 -1] range

**Efficiency constraints**

* your function is expected to print the result in less than 2 seconds

**Example**

|  |  |
| --- | --- |
| **Input** | **Output** |
| v: 5, 4, 3, 2, 8 | 3 |

Challenge 4: Deviation

Given an array of integer elements and an integer *d* please consider all the sequences of *d* consecutive elements in the array. For each sequence we compute the difference between the maximum and the minimum value of the elements in that sequence and name it the deviation.

**Your task is to**

* write a function that computes the maximum value among the deviations of all the sequences considered above
* print the value the standard output (stdout)

Note that your function will receive the following arguments:

* **v**
  + which is the array of integers
* **d**
  + which is an integer value giving the length of the sequences

**Data constraints**

* the array will contain up to 100,000 elements
* all the elements in the array are integer numbers in the following range: [1, 231 -1]
* the value of *d* will not exceed the length of the given array

**Efficiency constraints**

* your function is expected to print the result in less than 2 seconds

**Example**

|  |  |
| --- | --- |
| **Input** | **Output** |
| v: 6, 9, 4, 7, 4, 1 d: 3 | 6 |

**Explanation**

The sequences of length 3 are:

* 6 9 4 having the median 5 (the minimum value in the sequence is 4 and the maximum is 9)
* 9 4 7 having the median 5 (the minimum value in the sequence is 4 and the maximum is 9)
* 7 4 1 having the median 6 (the minimum value in the sequence is 1 and the maximum is 7)
* The maximum value among all medians is 6

Challenge 5: Maximum Difference

Given an array of integer elements, a subsequence of this array is a set of consecutive elements from the array (i.e: given the array v: [7, 8, -3, 5, -1], a subsequence of v is 8, -3, 5)

**Your task is to**

* write a function that finds a left and a right subsequence of the array that satisfy the following conditions
  + the two subsequences are unique (they don't have shared elements)
  + the difference between the sum of the elements in the right subsequence and the sum of the elements in the left subsequence is maximum
* print the difference to the standard output (stdout)

Note that your function will receive the following arguments:

* **v**
  + which is the array of integers

**Data constraints**

* the array has at least 2 and at most 1,000,000 numbers
* all the elements in the array are integer numbers in the following range: [-1000, 1000]

**Efficiency constraints**

* your function is expected to print the result in less than 2 seconds

**Example**

|  |  |
| --- | --- |
| **Input** | **Output** |
| v: 3, -5, 1, -2, 8, -2, 3, -2, 1 | 15 |

**Explanation**

The left sequence is : -5, 1, -2 and the right sequence is: 8, -2, 3.