Second GreatLow

Have the function `ArrayChallenge(arr)` take the array of numbers stored in `arr` and return the second lowest and second greatest numbers, respectively, separated by a space. For example: if `arr` contains [7, 7, 12, 98, 106] the output should be **12 98**. The array will not be empty and will contain at least 2 numbers. It can get tricky if there's just two numbers!

Examples

```
Input: [1, 42, 42, 180]
Output: 42 42
```

```
Input: [4, 90]
Output: 90 4
```

== TEST CASES ==

== INPUT ==
[1, 42, 42, 180]

<< EXPECTED OUTPUT: 42 42 >>

== INPUT ==
[4, 90]

<< EXPECTED OUTPUT: 90 4 >>

Most Free Time

Have the function `ArrayChallenge(strArr)` read the **`strArr`** parameter being passed which will represent a full day and will be filled with events that span from time X to time Y in the day. The format of each event will be **hh:mmAM/PM-hh:mmAM/PM**. For example, **`strArr`** may be ["10:00AM-12:30PM","02:00PM-02:45PM","09:10AM-09:50AM"]. Your program will have to output the longest amount of free time available between the start of your first event and the end of your last event in the format: **hh:mm**. The start event should be the earliest event in the day and the latest event should be the latest event in the day. The output for the previous input would therefore be **01:30** (with the earliest event in the day starting at **09:10AM** and the latest event ending at **02:45PM**). The input will contain at least 3 events and the events may be out of order.

Examples

```
Input: ["12:15PM-02:00PM","09:00AM-10:00AM","10:30AM-12:00PM"]
Output: 00:30
```

```
Input: ["12:15PM-02:00PM","09:00AM-12:11PM","02:02PM-04:00PM"]
Output: 00:04
```

== TEST CASES ==

== INPUT ==
["12:15PM-02:00PM","09:00AM-10:00AM","10:30AM-12:00PM"]

<< EXPECTED OUTPUT: 00:30 >>

== INPUT ==
["12:15PM-02:00PM","09:00AM-12:11PM","02:02PM-04:00PM"]

<< EXPECTED OUTPUT: 00:04 >>