Bookmark

Problem 4 - Part 1

Write a function called `getSublists`, which takes as parameters a list of integers named `L` and an integer named `n`.

- assume `L` is not empty
- assume `0 < n <= len(L)`

This function returns a list of all possible sublists in `L` of length `n` without skipping elements in `L`. The sublists in the returned list should be ordered in the way they appear in `L`, with those sublists starting from a smaller index being at the front of the list.

Example 1, if `L = [10, 4, 6, 8, 3, 4, 5, 7, 7, 2]` and `n = 4` then your function should return the list
`[[10, 4, 6, 8], [4, 6, 8, 3], [6, 8, 3, 4], [8, 3, 4, 5], [3, 4, 5, 7], [4, 5, 7, 7], [5, 7,`

Example 2, if `L = [1, 1, 1, 1, 4]` and `n = 2` then your function should return the list `[[1, 1], [1, 1], [1, 1], [1, 4]]`

Your function does not have to be recursive. Do not leave any debugging print statements when you paste your code in the box.

```
1 def getSublists(L, n):
2
```

Unanswered

You have used 0 of 10 submissions

Problem 4 - Part 2

Write a function called `longestRun`, which takes as a parameter a list of integers named `L` (assume `L` is not empty). This function returns the length of the longest run of monotonically increasing numbers occurring in `L`. A run of monotonically increasing numbers means that a number at position `k+1` in the sequence is either greater than or equal to the number at position `k` in the sequence.

For example, if `L = [10, 4, 6, 8, 3, 4, 5, 7, 7, 2]` then your function should return the value `5` because the longest run of monotonically increasing integers in `L` is `[3, 4, 5, 7, 7]`.

You may find it useful to use the function `getSublists` as defined above but you do not have to use this function. If you do use `getSublists`, the graders will use our implementation for `getSublists`. In the box for this problem, only paste the definition for the function `longestRun`.

Hint if you are Using `getSublists`

You may find the docstring for `range` useful.

`range(start, stop, step)`
The arguments must be integers. If the step argument is omitted, it defaults to 1. If the start argument is omitted, it defaults to 0. You cannot omit the start argument if you are specifying step.

An example you may find useful for this question:

```
>>> range(5, 0, -1)
[5, 4, 3, 2, 1]
```

Your function does not have to be recursive. Do not leave any debugging print statements when you paste your code in the box.

```
1 def longestRun(L):
2
```

Unanswered

You have used 0 of 10 submissions


Download on the
App Store

ANDROID APP ON
Google play