DS 5010 Homework 2

Instructions

- Submit your solutions on Canvas by the deadline displayed online.
- Your submission must include a single Python module (file with extension ".py") that includes all of the code necessary to answer the problems. All of your code should run without error.
- Problem numbers must be clearly marked with code comments. Problems must appear in order, but later problems may use functions defined in earlier problems.
- Functions must be documented with a docstring describing at least (1) the function's purpose, (2) any and all parameters, (3) the return value. Code should be commented such that its function is clear.
- All solutions to the given problems must be your own work. If you use third-party code for ancillary tasks, you **must** cite them. (You may use code from class notes without citation.)
- You may use functions from built-in modules (e.g., `math`). You may **NOT** use external modules (e.g., `numpy`, `pandas`, etc.).

Problem 1 Define a function called `median(x)` that satisfies the following criteria:

- Calculates and returns the median value of an iterable `x`
- The median is the "middle" value of a sorted list of numbers
- If the length of the list is even, then the median is the average of the two "middle"-most values
- You may assume `x` contains only numeric elements

Examples:

```
In : median([1, 2, 3, 4, 5])
Out: 3
In : median([1, 2, 3, 4, 5, 6]) # (3 + 4) / 2
Out: 3.5
```

Hint: You may find the `math.floor()` and `math.ceil()` useful.

Problem 2 Define a function called `iqr(x)` that satisfies the following criteria:

- Calculates and returns the interquartile range (IQR) of an iterable `x`
- The IQR is the difference (Q3 - Q1) where Q1 and Q3 are the first and third quartiles
- In an even-length list of $2n$ numbers, Q1 and Q3 are the median of the smallest n numbers and the median of the largest n numbers, respectively
- In an odd-length list of $2n + 1$ numbers, Q1 and Q3 are the median of the smallest $n + 1$ numbers and the median of the largest $n + 1$ numbers, respectively
- A dataset with one or fewer numbers has an IQR of 0
- You may assume `x` contains only numeric elements

Examples:

```
In : iqr([1, 2, 3, 4, 5]) # Q1 = 2, Q3 = 4
Out: 2.0
In : iqr([1, 2, 3, 4, 5, 6])  # Q1 = 2, Q3 = 5
Out: 3.0
```

Problem 3 Define a function called `fivenum(x)` that satisfies the following criteria:

- Calculates and returns Tukey's five number summary of an iterable `x`
- The five number summary is a `list` of five summary statistics: the minimum, Q1, the median, Q3, and the maximum of the dataset
- You may assume `x` contains only numeric elements

Examples:

```
In : fivenum([1, 20, 30, 45, 50, 60])
Out: [1, 20, 37.5, 50, 60]
```

Problem 4 Define a function called `order(x)` that satisfies the following criteria:

- Returns a `list` giving the indices of the elements of `x` as sorted in order from least to greatest
- That is, the first element of the output is the offset of the smallest element of `x`; the second element of the output is the offset of the second-smallest element of `x`; the last element of the output is the offset of the largest element of `x`, etc.
- In the event of ties, the original list order should be preserved
- The function should have no side-effects (i.e., it should not modify its input `x`)

Examples:

```
In : order([1, 99, 2, 100, -99])
Out: [4, 0, 2, 1, 3]
In : order([1, 99, 2, 100, -99, 100])
Out: [4, 0, 2, 1, 3, 5]
```

Problem 5 Define a function called `rank(x)` that satisfies the following criteria:

- Returns a `list` giving the sample ranks of the corresponding elements of `x`
- That is, the first element of the output is the rank of the first element of `x`; the second element of the output is the rank of the second element of `x`; the last element of the output is the rank of the largest element of `x`, etc.
- The rank is the ordinal index (1st, 2nd, 3rd, . . .) of an element when the dataset is sorted from least to greatest
- In the event of ties, the original list order should be preserved
- The function should have no side-effects (i.e., it should not modify its input `x`)

Examples:

```
In : rank([1, 99, 2, 100, -99])
Out: [2, 4, 3, 5, 1]
In : rank([1, 99, 2, 100, -99, 100])
Out: [2, 4, 3, 5, 1, 6]
```