22C:16 Practice Problem Set 6

Morning Section: Complete before Tuesday, 3-12-2013

Evening Section: Complete before Monday, 3-11-2013

These practice problems correspond roughly to the material covered in Week 7 (3/4-3/8). Every problem requires you to write a small (6-7 lines max) function. Try to write these functions using only what has been discussed in class (i.e., lectures and discussion sections).

1. Write a function called `concatenate` that takes a list of strings as a parameter and returns a long string that is the concatenation of all the strings in the list, taken in order. For example, if the given list is `["These", "are", "hello"]` then the function would return `"Thesearehello"`.

2. Write a function called `isSorted` that takes a list of numbers as a parameter and returns `True` if the list of numbers is sorted in ascending order and `False` otherwise. For example, if the given list is `[3, 8.5, 8.5, 11, 22]` then the function would return `True`; if the given list is `[3, 8.5, -11, 22]` then the function would return `False`.

3. Write a function called `subsetOf` that takes two lists and returns `True` if every element of the first list is also in the second list; otherwise the function returns `False`. For example, if the first list is `[3, 8.5, -22]` and the second list is `["hello", -22, "hi", 8.5, "goblin", 3]` then function would return `True`. On the other hand, if the first list is `[3, 8.5, -22]` and the second list if `["hello", -22, "hi", "goblin", 3]` then the function would return `False`.

4. Define a function called `gradeDistribution` that takes a list of exam scores and returns a list that contains the *distribution* of these scores. To be more precise let us suppose that the exam scores are out of 100 and therefore these are floating point numbers in the range 0 through 100 (inclusive of 0 and 100). The distribution of the scores we want you to compute is the number of scores that are in each of the ranges $[0, 10]$, $(10, 20]$, $(20, 30]$, $(30, 40]$, $(40, 50]$, $(50, 60]$, $(60, 70]$, $(70, 80]$, $(80, 90]$, and $(90, 100]$ (we are using $(A, B]$ to denote the range $A < s \leq B$ and $[A, B]$ to denote $A \leq s \leq B$). In other words, the first element in the list returned by your function should be the number of scores in the range $[0, 10]$, the second element should be the number of scores in the range $(10, 20]$, etc. You should use the following function header:

   ```
   def gradeDistribution(examScores):
   ```

5. Define a function called `upperCase` that takes as parameter a list of words and returns the sublist containing only those words that start with an upper case letter. By "word" we mean a string consisting entirely of lower or upper case letters. For example, if the parameter is `["hello", "Hi", "I", "ok", "besT"]` then the function should return `["Hi", "I"]`.

6. Define a function called `anotherUpperCase` that solves exactly the same problem as the previous one (that you solved using function `upperCase`), except that it modifies the given list in-place and does not return anything.

7. Define a function called `makeLower` that takes as parameter a word (i.e., a string with only letters) and returns the same string, but with all upper case letters not occurring in the first position turned into corresponding lower case letters. For example, if the parameter is `"AmaZinG"` then the function should return `"Amazing"`. Here is another example: if the parameter is `"gOOD"` the function should return `"good"`. Use the `ord()` and `chr()` functions that you recently learned about.

8. Write a function called `maxPairSum` that takes a list of numbers as a parameter and returns the pair of numbers in consecutive positions that add up to the largest value. For example, if the given list is `[3, -1, 4, 2, 5, -1, 11, -8]` then the function would return the list `[-1, 11]`.

9. Write a function called `minIndex` that takes a list of numbers as a parameter and returns the index of a smallest element in the list. If there are several smallest numbers in the list, it does not matter which index is returned. For example, if the parameter is `[3, -1, 2, 3, -1, 11]` then the function could return 1 or it could return 4.

10. Write a function called `moveMin` that takes a list of numbers as a parameter and moves the minimum element to the front of the list. You should assume that the list has a unique minimum elememnt. Also, your function needs to do this task in-place and make sure that the relative order of the remaining elements is unchanged. For example, if the list is `[3, -1, 2, 3, 11]`, then after the function is executed the list should change to `[-1, 3, 2, 3, 11]`. The function should call `minIndex` defined in the previous problem.
