CSCI 3675 – Principles of Programming Languages
Fall 2022
Homework 1 – Polymorphism, recursion, and data structures with Python
Due Sunday, September 4, at 11:59 PM

All the solutions for this assignment should be implemented in Python.

1. (10 pts) Write a function `halveEvensImperative`, using an imperative style of programming, that returns half of each even number in the list. For example, `halveEvensImperative ([0, 2, 1, 7, 8, 56, 17, 18])` should return `[0, 1, 4, 28, 9]`.

2. (10 pts) Write a recursive function `halveEvensRecursive`, that returns half of each even number in the list. For example, `halveEvensRecursive ([0, 2, 1, 7, 8, 56, 17, 18])` should return `[0, 1, 4, 28, 9]`.

3. (10 pts) Write a function `halveEvensComprehension`, using list comprehension, that returns half of each even number in the list. For example, `halveEvensComprehension ([0, 2, 1, 7, 8, 56, 17, 18])` should return `[0, 1, 4, 28, 9]`.

4. (10 pts) Write a function `capitalizeImperative`, using an imperative style of programming, which, given a word, will capitalize it. That means that the first character should be made uppercase and any other letters should be made lowercase. For example, `capitalizeImperative ('grEENVIlle')` should return 'Greenville'. Your definition should use the functions `upper` and `lower` that change the case of a character.

5. (10 pts) Write a function `capitalizeComprehension`, which, given a word, will capitalize it. That means that the first character should be made uppercase and any other letters should be made lowercase. For example, `capitalizeComprehension ('grEENVIlle')` should return 'Greenville'. Your definition should use a list comprehension and the functions `upper` and `lower` that change the case of a character.

Functions that work with several types are called polymorphic. Polymorphism can facilitate code reuse. For example, the built-in function `sum`, which adds the elements of a sequence, works as long as the elements of the sequence support addition. For the following three problems, you are not allowed to use this function.

6. (10 pts) Write a function `sumImperative`, using an imperative style of programming, that returns the sum of the numbers in a list. For example, `sumImperative ([0, 2, 1, 7, 8, 56, 17, 18])` should return `109`, and `sumImperative ([1, 2.0])` should return `3.0`.

7. (10 pts) Write a recursive function `sumRecursive`, that returns the sum of the numbers in a list. For example, `sumRecursive ([0, 2, 1, 7, 8, 56, 17, 18])` should return `109`, and `sumRecursive ([1, 2.0])` should return `3.0`.

8. (10 pts) Write a function `palindromeImperative`, using an imperative style of programming, that returns `True` if the given list is a palindrome (reads the same backward as forward), and `False` otherwise. For example, `palindromeImperative ([0, 2, 0])` should return `True`, and `palindromeImperative ('abb')` should return `False`. No other functions should be called from this function; use slicing operations instead.

9. (10 pts) Write a recursive function `palindromeRecursive`, that returns `True` if the given list is a palindrome (reads the same backward as forward), and `False` otherwise. For example, `palindromeRecursive ([0, 2, 0])` should return `True`, and `palindromeRecursive ('abb')` should return `False`. No other functions should be called from this function; use slicing operations instead.

10. (10 pts) Write a recursive function `inRangeRecursive`, that returns all numbers in the input list within the range given by the first two arguments (inclusive). For example, `inRangeRecursive (5, 10, range(1, 15))` should return `[5, 6, 7, 8, 9, 10]`.

11. (Extra Credit – 10 pts) Write a function `inRangeComprehension`, using list comprehension, that returns all numbers in the input list within the range given by the first two arguments (inclusive). For example, `inRangeComprehension (5, 10, range(1, 15))` should return `[5, 6, 7, 8, 9, 10]`.

Submission

- Submit your work as one `hwk1.py` file containing all your functions, via Canvas.
  - The file must be in a simple text format; do not submit Word, PDF, RTF, JPG, etc.
  - Also make sure that any auxiliary information (such as your name or question numbers) is commented out.