Extra Homework Assignments

The list of homework assignments below gives an opportunity to take credit for increased practice. Every correct solution to a homework problem below is worth 2 points. The deadline for handing in the answers is Wednesday 5 March, at 11AM, bring your answers to class.

1. The function \texttt{StretchWord(s)} returns a string where all characters in \( s \) are separated by one space. For \( s = "\text{hello}" \) \texttt{StretchWord(s)} returns "\( h \ e \ l \ l \ o \)". Write an iterative and a recursive version of \texttt{StretchWord}.

2. Write code to make a frequency table of words, stored as a binary tree. The empty tree is the empty tuple \((\text{}\text{}\text{}\text{})\). A general node in the tree is of the form \((\text{left}(\text{(word,n)}),\text{right})\), where \((\text{word,n})\) contains the string \( \text{word} \) and the number \( \text{n} \): \( \text{n} \) is the frequency of the word \( \text{word} \). The left and right are again trees. All words in left are lexicographically less than \( \text{word} \), all other words are in right. The tree \( T \) is created from a list of words by repeatedly invocation of \( T = \text{Update}(T, \text{word}) \). The function \texttt{Update} takes on input a tree \( T \) and a string \texttt{word}. \texttt{Update} then returns the updated frequency table.

3. The \texttt{Update} of the previous exercise is a recursive function. Use a stack to write an iterative version.

4. Consider a list of tuples \((\text{name}, \text{score})\), where \text{name} is a string and \text{score} a floating point number. Sort the tuples in the list along the scores. Tuples with highest score must appear first in the sorted list on return.

5. Write a more general implementation of \texttt{QuickSort} to sort elements in a list. In addition to the list, the general \texttt{QuickSort} takes on input a function \texttt{LessThan}. The \texttt{LessThan} function compares two elements of the type in the list to sort and returns \texttt{True} if the first element is less than the second, and \texttt{False} otherwise. Demonstrate the working of your function on lists that contain tuples \((\text{name}, \text{score})\) like in the previous exercise.

6. A grayscale picture is stored as a two dimensional numpy matrix. The \((i, j)\)-th entry contains the grayscale of the \((i, j)\)-th pixel. This grayscale is a short integer between 0 and 255. Write a fading function which replaces the \((i, j)\)-th pixel by averaging its grayscale with that of its eight neighbors.

7. Apply divide and conquer to compute the average of the elements in a numpy array. The average of the whole array is the average of the average of the first half and the average of the second half. To be memory efficient, do not take slices of an array: the function \texttt{Average(A,first,last)} returns the average of \( A\lfloor \text{first} : \text{last} \rfloor \).
8. A permutation of \( n \) elements is represented as a numpy array of length \( n \) with entries in the set \( \{0, 1, \ldots, n-1\} \). Every element of \( \{0,1,\ldots,n-1\} \) occurs exactly once in each permutation. The identity permutation is \([0 \; 1 \; \ldots \; n-1]\). Define a class Permutation. The constructor takes on input \( n \) and returns the identity permutation. Override the __mul__ operation to compute the multiplication of permutations, defined as composition of permutations. For example: \([2 \; 1 \; 3] \times [3 \; 2 \; 1] = [2 \; 3 \; 1]\) means swapping 2 and 1 after swapping 3 and 1.

9. A Sierpinski carpet starts with a rectangle \([a, b] \times [c, d]\) and repeatedly removes the middle rectangle. The piece that is removed in the first step is \([a + (b - a)/3, b - (b - a)/3] \times [c + (d - c)/3, d - (d - c)/3]\). Then the removal proceeds with the remaining pieces. Call the \( n \)-th Sierpinski carpet as what remains after \( n \) removals. Write a Python function to compute the total area of what is removed.

10. Build a GUI to draw a Sierpinski carpet on canvas, see the previous exercise for its definition.