Review of the lectures 1 to 15

The final exam will be open book. Calculators or laptop computers are not allowed. Good examples of questions are quizzes and homework assigned at the end of each lecture; and of course also the first midterm. The material is focused on the first 15 lectures (or the first five weeks of the course). The first five lectures gave a tour of Python (scripting, extending, numpy, OOP, and GUIs). The emphasis on the following ten lectures was then on recursive problem solving: algorithms, drawings, and data structures.

This sheet contains some preliminary examples of questions which may help you prepare your study of the first 15 lectures.

1. The function `DigitCount(n)` returns the number of digits of a natural number `n`.
   Its most simple implementation is `len(str(n))`.

   (a) Write a recursive definition of `DigitCount(n)`. (Do ***not*** use `str(n)`.)

   (b) Modify the recursive definition so it prints the stack of all recursive function calls.
   Use an accumulating parameter for the proper indentations.

   (c) Extend the recursive definition so the result is accumulated in a parameter.

   (d) Make a stack of the recursive function calls and then turn the recursive function into an iterative definition.

2. Consider a `numpy` 0/1 matrix `A`. Write a function which returns `(a,b,c,d)`. This tuple defines the block `A[a:b,c:d]`. All entries in `A[a:b,c:d]` are one.

3. Let `A` be matrix of characters. Each entry is either a space, a cross `X` or a circle `O`. Write a function `FourInRow(A,c)` which returns true if the character `c` occurs four in a row (horizontally, vertically, or diagonally) in the matrix `A`.

4. Consider the Koch flake, defined by two parameters: n and k. For $k = 0$, the Koch flake is a regular n-gon, spanned by n equispaced points on the unit circle.

   (a) Write a function to compute the total length of all sides of a regular n-gon with its n points evenly distributed on the unit circle $x^2 + y^2 = 1$.

   (b) Give Python code for a function to return the list of coordinates of all points for a Koch flake of level k starting from a regular n-gon.

   (c) Write a recursive function, with parameters n and k, to compute the length of the sides of the Koch Flake.

5. An arithmetical expression is stored in a binary tree as follows. The node of the tree contains one of the four operators: `+`, `-`, `*`, `/`, represented by one character. The left and right children are the operands. The operands can be numbers or again arithmetical expressions.

   (a) Describe how to use tuples to represent arithmetical expression trees. Give a nontrivial example: take an arithmetical expression with two levels of nested parentheses and write its representation as a tree.

(b) Write Python code for a function to generate a random arithmetical expression tree of depth at most k. Use a random number generator to generate the operators. The operands are random natural numbers between 1 and 9.

(c) Give a Python function which takes on input an arithmetical expression tree and returns a string representation of the tree. The string representation should be as ordinary humans would expect it to be (applying `str()` to the data structure will receive no credit).

(d) A Python function `eval(T)` returns the value of the arithmetical expression defined by the tree in `T`. Implement this function.

(e) Give code for a function to count the number of leaves.

(f) Consider an arithmetical expression tree where the operands are natural numbers. These natural numbers are indices in a list `X` of variables. Write a function to replace the numbers `n` of all leaves by the symbols in a list `X`, i.e.: `n` is replaced by `X[n]`.

6. An algebraic expression involves only `+` and `*` operations, i.e.: we exclude divisions and `-` is implemented via multiplication with -1. A recursive data structure of an algebraic expression consists of an operator and a list of operands. The operands are added or multiplied, depending on the type of operator. The operands can again be algebraic expressions.

   (a) Describe how you would represent an algebraic expression in Python. Give nontrivial examples of algebraic expressions with their corresponding recursive data structure.

   (b) Write code to generate random algebraic expressions with integer operands as their leaves, and of prescribed maximal depth.

   (c) Give the code of a function to evaluate an algebraic expression.

7. Consider an interval $[a, b]$ and a list of nonoverlapping subintervals. The list is represented as a list of tuples. Each tuple contains two elements: the left and right end of each subinterval.

   (a) Define a function `RandomSubs(a,b,n)` which returns a list of `n` nonoverlapping subintervals of the interval $[a, b]$. The length of each subinterval varies uniformly between $(b-a)/(20n)$ and $(b-a)/(10n)$. Generate the list by picking the left bounds in increasing order. Apply `random.shuffle()` before returning the list.

   (b) Give code to sort a list of nonoverlapping subintervals applying mergesort.

   (c) Give code to sort a list of nonoverlapping subintervals applying quicksort.

   (d) Given a list of sorted nonoverlapping subintervals and a number `x`. Write code for a function to decide whether `x` belongs to one of the subintervals in the list. For efficiency, you must use binary search.

8. Consider two sets of characters of equal length. The second set `V` contains five vowels: `a`, `e`, `i`, `o`, and `u`. The first set `N` contains five nonvowels. Give a Python script to write all words of combinations of ten letters of `N` and `V`. Each character occurs exactly once and each nonvowel is followed by a vowel. You must use recursion, nesting loops five levels deep is not allowed.

The final exam is in 208 Burnham Hall on Thursday 8 May from 8:00 to 10:00AM.