Information about the Computational Thinking Exam

The exam is an **open book exam**. You can bring whatever you like to the exam, apart from communication devices and electronics. In other words, you do not have to memorize the details of a proof, or remember all properties of a particular model. However, this also means that we will hardly ask questions that demand you to list/reproduce something in the script or in the exercises. So most questions will instead challenge you if you *understand* the key concepts.

The exam will have **three kinds of questions**: Multiple choice questions, typical questions, special questions. You can expect that most of the points will come from multiple choice and typical questions. That means you can pass, with a good grade, if you learn these parts well.

A now defunct lecture (called TI2) had some topical **overlap** with this lecture: Chapter 4 (Data and Storage) was taught pretty much 1:1 already in TI2. Chapter 3 (Cryptography) was also partially taught in TI2. CoTi teaches a bit more crypto than TI2, so you should be able to answer TI2 crypto exam questions. We collected all these questions from the previous exams here.

1. Multiple Choice Questions

Multiple choice questions are generally a bit simpler than the other two types. They are also the only questions where you do not need to justify your answer. While some **clicker** questions do hardly make sense in the context of an exam (we used them to motivate the next part of the lecture), most actually may be asked in an exam as well. Some may be a bit difficult in an exam (e.g. the questions about the early states in 6.37 Study or Party), and some may be a bit easy (e.g. 7.1 Halting Problem Runtime), but surprisingly most of these clicker questions are at the right level of difficulty for a multiple choice question.

2. Typical Questions

Some questions come in a typical format. The goal of these questions is to test a core skill that you should have learned in this class. For each chapter you can expect the following templates. Note that there will not be a question for every Chapter in every exam.

1. Algorithms
    a. We describe a problem and ask you to provide a Python-like algorithm. The problem will allow you to solve it with one of the techniques from Chapter 1 (recursion, backtracking, dynamic programming or linear programming). We will not punish any Python syntax mistakes as long as the intention was clear.
    b. We give you some piece of Python code, and ask you what it does, or what is wrong with it.
    c. We ask you to analyze a given Python program and ask you about its runtime, and how to improve the runtime.
2. Complexity
    a. Reductions: find a polynomial reduction from a problem to another problem (one of these problems may or may not be known already).

    b. Approximations: find an approximation algorithm, or show that some given approximation algorithm works.
    c. Find a bad counterexample for a given approximation algorithm,
3. Cryptography
    a. Argue about the security of a new protocol. Why does it (not) work? We may ask for a formal proof, or also for informal arguments.
    b. Modify a given protocol to satisfy higher notions of security, and argue why.
    c. Identify the combination of cryptographic primitives that can solve a particular given problem.
    d. Show that a given protocol implements a specific cryptographic primitive, i.e., argue that the protocol has all the properties of the primitive. For instance, Pedersen's protocol implements a commitment scheme.
4. Databases
    a. Given a database schema, write the SELECT statement based on a natural language question.
    b. Given a database schema, translate a SELECT statement back into natural language.
    c. Given a database schema, find errors in a given SELECT statement.
5. Machine Learning
    a. Given some data, suggest a model to fit to this data, or judge the fit of a model we give you. Discuss bias, variance, how you expect the weights to look like.
    b. Derive a gradient descent update rule for a given loss function. Discuss how a regularization term affects the model.
    c. Given a function, for what initial values does gradient descent find the global minimum.
    d. Given a decision tree splitting criterion, build a small decision tree. Discuss how different splitting criteria affect the decision tree built.
6. Neural Networks
    a. Given a network with weights (also CNN, RNN, Attention): What does the network compute? Perhaps, the model might have some weight errors that need to be fixed.
    b. Compute and evaluate the weight changes of one step of back propagation in neural networks
    c. What weights would you choose in a model in order to achieve a certain goal?
7. Computability
    a. TM (or related model: like Tiling), what does a given "code" (states, transitions) do. Potentially find errors in code.
    b. Given some computing model (possibly also a new model not seen in class), can it simulate another model such as a TM.
    c. Evaluate the difficulty of variants of the PCP problem (or some other problem): What is polynomial, computable but hard, or undecidable, and why?

3. Special Questions

There will be questions which do not fit into the list of typical questions. A question could modify an existing algorithm/theorem/model by removing or adding assumptions and asking you what this changes. Another possible kind of question is giving you an alternative “solution” to one problem and you have to find out if this is actually a solution. Another kind of question could ask you to connect some ideas across chapters. You will do well in these kinds of questions if you understand all seen concepts (lecture, lecture notes, notebooks, exercises).