CS 581: Theory of Computation
James Hook
Final exam.

This is a closed-notes, closed-book exam.

1. True, False, or Open
   (a) P = NP
   (b) In a reasonable proof system all provable things are true.
   (c) In a reasonable proof system all true things are provable.
   (d) All regular sets are finite.
   (e) All Turing-decidable sets are Turing-recognizable.
   (f) The intersection of a context free language and a regular language is a regular language.

2. What is a verification problem? How do verification problems relate P and NP? Give an example.

3. Sipser presents two theories of arithmetic, $Th(N, +)$ and $Th(N, +, \times)$. This problem focuses on the weaker theory $Th(N, +)$.
   (a) Sipser uses the "prolog style" predicates to represent arithmetic operations. For each formula: (1) identify it as either an open formula (in which case please list the free variables) or a sentence and (2) describe what the formula means.
      i. $+(1, 2, 3)$
      ii. $+(y, y, x)$
      iii. $\exists y. + (y, y, x)$
      iv. $\forall x.\forall y.\exists z. + (x, y, z) \wedge +(y, x, z)$
   (b) Sipser shows that $Th(N, +)$ is decidable. In that construction how does Sipser represent the meaning of an atomic formula? What results justify this choice? (you may refer to theorems from reading, lecture or homework)
   (c) How does Sipser represent the meaning of the connective $\neg$ (negation)? What results justify this choice?
   (d) How does Sipser represent the meanings of the connectives $\wedge$ and $\vee$ (and and or)? What results justify this choice?
   (e) How does Sipser represent the meaning of the $\exists$ quantifier? What results justify this choice?
   (f) To what decidable language property does Sipser reduce the truth of statements in $Th(N, +)$?

4. Lambda calculus.

   Curry's combinatory logic uses a set of combinators to capture the behavior of a typed subset of the λ-calculus. The two primary workhorse combinators are S and K, given below:

$$S = \lambda x.\lambda y.\lambda z.(xz)(yz) \tag{1}$$
$$K = \lambda x.\lambda y.x \tag{2}$$

   One elementary fact that can be shown by calculation is that the identity function ($\lambda x.x$) can be implemented by SKK. Prove this fact by reduction in the lambda calculus. That is prove that:

$$((\lambda x.\lambda y.\lambda z.(xz)(yz))(\lambda x.\lambda y.x))(\lambda x.\lambda y.x)$$

   reduces to $\lambda x.x$. (The calculation is more compact if you do not expand the instances of K until you need to.)

5. Two forms of reduction were defined formally using a function to implement the reduction: mapping reducibility and polynomial time reducibility.

   (a) Sketch the common framework of the two definitions

   (b) Discuss the different requirements on the reduction function, f, in these definitions.

   (c) Give at least one theorem or lemma for each reduction technique that allows it to be used in a reduction argument.

   (d) Describe how each of these theorems are used in an argument. (You can give a very high level sketch of the argument, but be as precise as possible about how the result being illustrated is used.)

6. You attend a talk. The speaker claims to have developed the ultimate optimizing compiler. Given any program the compiler generates the shortest assembly code that implements the program. Being a well trained computer scientist you are skeptical.

   (a) If you assume the claims are correct, what can you conclude about the programming language compiled? Why?

   (b) If the speaker argues that the compiler correctly compiles programs of arbitrary complexity in a general purpose programming language, identify at least one result studied in this class that contradicts the speaker's claims.