1. (**15 points**) Let boolean halt(String src, String input) be a Java method that is supposed to return true if the Java program with the source code given by string src halts when run with input input and returns false otherwise. We showed in class that it is impossible to write such a method that will work for all strings src and input. Now, we will add one more restriction: halt fails (i.e. throws an exception) if the string for input is the source code for a Java program. For example, halt could run the Java compiler on input and if it compiles successfully, then halt throws a InputWrittenByASmartAleckException. In all other cases, halt should correctly answer whether the program given by src halts when run with input input.

   Show that even with this restriction, it is impossible to write a method halt that returns true if the program described by src halts when run with input input and returns false otherwise.

2. (**15 points**) In class, we constructed one example that must cause a proposed function for halt to give the wrong answer or never terminate. Show that for any proposed implementation of halt there must be an infinite number of inputs that cause it to give the wrong answer or never terminate.

3. (**35 points**) Download the program mystery.java from

   http://www.ugrad.cs.ubc.ca/∼cs421/hw/7/mystery.java

   Look over the code, compile it, and run it – I promise that it's not malicious.

   1. (**5 points**) What does the program do? Just give a one-sentence description of the output that it produces. You'll get to explain *how* it does it in the rest of the question.
   2. (**5 points**) What is string s for?
   3. (**5 points**) What does method x() do?
      A one sentence answer is enough. You'll get to explain the details in the next three questions.
   4. (**5 points**) What do the first four buf.append(. . .)'s in x() do?
   5. (**5 points**) What does the first for loop in x() do?
   6. (**5 points**) What does the second for loop in x() do?
   7. (**5 points**) What does method fix(String) fix?

4. (**20 points**, Sipser Problem 3.9)
   Let a k-PDA be a pushdown automaton that has k stacks. Thus a 0-PDA is an NFA and a 1-PDA is a conventional PDA. You already know that 1-PDAs are more powerful (recognize a larger class of languages) than 0-PDAs.

   1. (**10 points**) Show that 2-PDAs are more powerful than 1-PDAs.
   2. (**10 points**) Show that 3-PDAs are not more powerful than 2-PDAs.
      (Hint: Simulate a Turing machine tape with two stacks.)

5. (**20 points**, Sipser Problem 3.11)
   A *Turing machine with doubly infinite tape* is similar to an ordinary Turing machine, but its tape is infinite to the left as well as to the right. The tape is initially filled with blanks except for the portion that contains the input. Computation is defined as usual except that the head never encounters an end to the tape as it moves leftward. Show that this type of Turing machine recognizes the class of Turing-recognizable languages.

(continued on next page)

6. (**30 points**, Sipser Problem 3.14)

A *queue automaton* is like a push-down automaton except that the stack is repoaced by a queue. A *queue* is a tape allowing symbols to be written only on the left-hand end and read only at the right-hand end. Each write operation (we'll call it a *push*) adds a symbol to the left-hand end of the queue, and each read operation (we'll call it a *pull*) reads and removes a symbol at the right-hand end. As with a PDA, th einput is placed on a separate read-only input tape, and the head on the input tape can only move from left to right. The input tape contains a cell with a blank symbol following the input, so that the end of the input can be detected. A queue automaton accepts its input by entering a special accept state at any time. Show that a language can be recognized by a deterministic queue automaton iff the language is Turing recongnizable.