

Due: Thursday 5/8/08 at 11:30AM (beginning of class)

Only Problem 7 requires an electronic submission: call the file containing your Turing machine `add.jff`, and deposit it at the top-level of your dropbox. You must also submit a printout of your Turing machine.

All other problems require written solutions. You must submit these in hard copy. Feel free to type as many of your answers as you possibly can. Whatever you end up writing or drawing by hand must be legible. **Points will be taken off for sloppy, hard-to-read, vague, or unclear solutions.**

1. (10 points) Solve problem 8 on page 562. As always, you must provide a complete and unassailable proof.
2. (10 points) Solve problem 9 on page 562. Hint: Use a proof by cases.
3. (10 points) Solve problem 14 on page 564. If your answer is yes, then your proof must consist of the shortest code word that proves it, accompanied by a drawing of the corresponding TM.
4. (10 points) Solve problem 2 on page 590. Hint: Use a proof by cases.
5. (10 points) After studying the type 0 grammar given above problem 1 on page 590 and convincing yourself that it generates all the words containing an equal number of a 's and b 's, modify it to obtain a grammar that generates the language *MOREA* defined on page 205. For full credit, your solution must MINIMIZE the number of changes to the grammar. Hint: There is one solution in which only two productions are modified or added.
6. (10 points) Solve problem 12 on page 592. Do not forget to prove your answer.
7. (10 points) Solve problem 2(ii) on page 617 with a standard deterministic 1TM (possibly using the STAY option). You may assume that all the numbers to be added are positive. It does not matter where the tape head is pointing when the string is accepted and the correct output is produced. You may use the INSERT building block at most once. For full credit, keep your TM as small as possible. Hint: My solution contains 6 states.
8. (10 points) Prove that, if $P = NP$, then $NP = \text{co-NP}$.
9. (10 points) Prove that if $P = NP$, then a polynomial time algorithm exists to produce a satisfying assignment for a given satisfiable boolean formula. Note that solving the SAT problem does *not* produce a satisfying assignment since an answer to the SAT problem is simply 'yes' or 'no', depending on whether such an assignment exists.
10. (10 points) Prove that, if any *one* NP-complete problem can be solved in polynomial time, then *every* problem in NP has a polynomial time solution, that is, $P = NP$.