CS 39 Fall 2005 Theory of Computation	Final Exam December 2005 *Attach this sheet to your submission*	Prof. Amit Chakrabarti Computer Science Department Dartmouth College

I started this exam at time ______________ on date ______________.

I am submitting this exam at time ______________ on date ______________.

I pledge my honor that the times and dates I have reported above are accurate to the best of my knowledge.

Your signature: ____________________

Your name: ____________________

Important Points to Note

- This exam is due **48 hours** from when you first look at it, or on **December 6, at 6:00pm sharp**, whichever comes earlier. Submit it into Chakrabarti's mailbox.
- There are **6 sections** for a total of **100 points**.
- Start each problem (or part thereof) on a fresh page. Use **at most ten pages** (that's five sheets, letter or A4 size) for the entire exam. You will be held to this limit, strictly.
- The exam is open book and open notes. You may use, without proof, any results proved in class or in Sipser's book, provided you state clearly what result(s) you are using.
- You may not discuss this exam in any detail whatsoever with anyone, even after you have submitted it. If you have questions on the course's material, please ask the professor or the TA; we are willing to help you to any extent with understanding of the material.
- If you are completely stuck on a problem, **you may ask for a hint** which will be provided via email. Your score for that problem will then be halved. Please note that the hints have been prepared in advance and cannot be customized. To ask for a hint, send email to Amit Chakrabarti (`ac@cs`) **as well as** Khanh Do Ba (`kdb@cs`) saying something like "I'd like the hint for Problem 6.1 and Problem 2." We will try to get back to you ASAP but please remember that we are humans too and we must sleep.
- Good luck!

Section	**Points**	**Score**
1	15	
2	10	
3	20	
4	10	
5	20	
6	25	
Total	**100**	

1. For a string $x \in \{0,1\}^*$, let $\beta(x)$ denote the value of x when interpreted as a binary number. Thus, $\beta(1) = 1$, $\beta(100) = 4$, $\beta(001011) = 11$, and so on. Prove or disprove that the language $\{x \in \{0,1\}^* : \beta(x) \text{ is a perfect square}\}$ is regular.

   [15 points]

2. For a string $x \in \{0,1\}^*$, let $N_0(x)$ and $N_1(x)$ denote the number of 0's and 1's (respectively) in the string x. Prove or disprove that the language $\{x \in \{0,1\}^* : x \text{ does not have a substring } y \text{ such that } |N_0(y) - N_1(y)| = 3\}$ is regular.

   [10 points]

3. You are given a black box that decides membership in a certain language $L \subseteq \Sigma^*$ (for some alphabet Σ) as follows: you can feed it a query string $x \in \Sigma^*$ and ask it "Is $x \in L$?" The black box replies with either "Yes" or "No." Since it is a *black* box, you have no way of finding out how the device works. Your task is to decide whether or not $L = \emptyset$ by using a *finite* sequence of queries to this black box.

   As stated, the problem cannot be solved, because for any integer r, if you ask r queries and receive only "No" answers from the black box, you are unable to decide whether or not $L = \emptyset$: you haven't yet found an element in L, but perhaps the very next query would have yielded a "Yes" answer? However, suppose a little birdie tells you that L is generated by a Chomsky Normal Form CFG with n variables, but doesn't tell you what the CFG is. Prove that you can now solve the problem by asking no more than $r(n)$ queries, for a certain integer $r(n)$, depending on n. In your solution, give an explicit formula for whatever $r(n)$ you use.

   [20 points]

4. Let E_1 and E_2 be enumerator Turing machines (not necessarily lexicographical) over the same alphabet Σ. Give a *direct* construction of an enumerator for the language $\mathcal{L}(E_1) \cap \mathcal{L}(E_2)$. By "direct," we mean that you are *not allowed* to use the theorems connecting enumerators with recognizers.

   [10 points]

5. A context-free grammar over an alphabet Σ is said to generate "almost all strings" if it generates all but a finite number of strings in Σ^* (this finite number may be zero). Consider the computational problem of determining whether a given CFG generates almost all strings. Either describe an algorithm to solve the problem or prove, via undecidability, that no such algorithm exists.

   [20 points]

6. Let HALFCLIQUE $= \{\langle G \rangle : G \text{ is an undirected } (2n)\text{-vertex graph and has a clique of size } n\}$.

   6.1. Prove that HALFCLIQUE is NP-complete.

   [15 points]

   6.2. Suppose you are given a magic black box that somehow decides HALFCLIQUE in polynomial time. Prove that, using this black box, you can solve the following computational problem in polynomial time: given an undirected $(2n)$-vertex graph G, output a subset of n vertices of G that form a clique, or else report that no such subset exists.

   [10 points]