Introduction to Algorithms	**Problem Set 9 (1 problem/5 subproblems)**
CS 4820 Fall 2018	**Due 11:59pm Thursday, November 15**

Your homework submissions need to be typeset (hand-drawn figures are OK). See the course web page for suggestions on typing formulas.

The solution to each question need to be uploaded to CMS as a separate pdf file. To help provide anonymity in your grading, do not write your name on the homework (CMS will know it's your submission). (Questions with multiple parts need to be uploaded as a single file.)

To prove a problem is decidable, you must give an algorithm that decides the problem and prove its correctness. **To prove a problem is undecidable,** you can use either a reduction from an undecidable problem to your problem or Rice's Theorem (if Rice's Theorem applies).

Proving computability. For each of the following problems, prove whether it is decidable.

(a) *(6 points)* MODIFIES INPUT. Given a deterministic program M and an input x already in memory, does $M(x)$ ever modify x in place? (This is a relevant question for security: you might want to make sure someone can't write to a protected part of memory.)

(b) *(6 points)* LIMITED MEMORY HALTING. Given a deterministic program M consisting of n instructions, an input x, and an integer c, does $M(x)$ halt having used less than c bits of memory total (where memory is rewriteable)?

(c) *(6 points)* PROGRAM AGREEMENT. Given two deterministic programs M and M' and one input x, do M and M' agree on input x - i.e. do they both accept, both reject, or both never terminate?

(d) *(6 points)* ACCEPTS 1 IN INPUT. Given a deterministic program M, does M accept at least one input x that has a 1 in the input string?

(e) *(6 points)* QUANTIFIED SAT. Deciding who has a winning strategy in games offer an interesting class of hard problems. Here we consider a very CS-ish game: two players play setting variables in a formula: Player 1 wants to make the formula true, player 2 wants to make it false, and they get to set variables in turn. Here is how the game is played: given a formula Φ with variables $x_1, \ldots, x_n$; player 1 sets variable x_1, then player 2 sets variable x_2, etc. In this game, player 1 has a winning strategy if he can set variable x_1 in a way, that no matter how player 2 sets variable x_2, he then can then set variable x_3, and so on $\ldots$, so that the formula becomes true. The problem of deciding if player 1 has a winning strategy is formalized by the following QUANTIFIED SAT problem.

Consider a SAT formula Φ (in conjugative normal form) with variables $x_1, \ldots, x_n$ with an even n. The QUANTIFIED SAT problem is to decide if the following quantified version of the formula is true

$$\exists x_1 \, \forall x_2 \ldots \exists x_{n-1} \, \forall x_n \, \Phi$$

meaning that there is a setting fo x_1 such that for all settings of x_2, there is a setting of x_3, etc. that makes the formula true. For an example, for the formula

$$\Phi_1 = (x_1 \vee x_2) \wedge (\bar{x}_2 \vee x_3) \wedge (x_2 \vee \bar{x}_3) \wedge (x_4 \vee \bar{x}_4)$$

the quantified version is true: by setting $x_1 = T$, no matter what x_2 is set, there is a way to satisfy the formula. In contrast for the formula

$$\Phi_2 = x_1 \wedge (x_2 \vee x_3) \wedge (\bar{x}_3 \vee x_4)$$

is not true: we need to set $x_1 = T$, now if $x_2 = F$, then we must set $x_3 = T$, and now setting $x_4 = F$ gets the formula false.