Because we will hand out sample solutions to this problem set in the last class (11/28/2018), this homework cannot be submitted later than at the beginning of class on Wednesday, 11/28/2018.

(1) Describe at least one way in which what you have learned in this class helps you gain a new perspective on your own research area, or gives you new tools which you think will help improve your own research. You can talk about broad understanding, or about a specific problem for which you think you might now have better approaches available, or anything else relating to your own work. If you are not research-active, then talk about perspectives on another area of CS you are most interested in.

Your answer to this question should reflect some serious thought, not superficial comments that you come up with in 5 minutes. Also, please submit your answer to this question on a separate piece of paper, since David will “grade” it.

(2) [10 points]
Problem 13.6 from the textbook.

(3) [10 points]
Problem 13.15 from the textbook.

(4) [10 points]
Problem 13.17 from the textbook.

(5) [10 points]
Selecting Profitable Projects: In class, we discussed the online problem of selecting profitable projects/activities. It is pretty easy to see that in the version we discussed, no good competitive ratio could be achieved. Here is a different variant: At the end of every day $d$, you receive some job offers (zero or more). Each job offer is a pair $(v_i, t_i)$. $v_i$ is the value of the offer, and $t_i > d$ is the deadline. This means that if you do job $i$ any day on or before the deadline $t_i$, you obtain $v_i$ dollars. If you don’t do the job, or do it after the deadline, you get 0 dollars for it. Each job takes exactly an entire day to do, so each day, from among all the non-expired jobs, you get to choose one. Of course, once you did (and profited from) a job, you can’t do the same job again another day. This being an online problem, you don’t know on day $d$ what new job offers you will receive at the end of day $d$, or in future days.

Probably the simplest online algorithm for this problem is the greedy algorithm: among all jobs available on day $d$, do the one with highest profit. That is, ignore deadlines in the decisions. Prove that the greedy algorithm is (at least) $\frac{1}{2}$-competitive for this problem.

(6) [10 points]
Space Stations: Suppose that you play the following simple game where you defend $n$ space stations against an attacker. Each space station starts with energy $c$. The game is played in rounds. In each round, the following happens:

(a) First, an attacker chooses an attack vector $a$ of how to attack your space stations. The attack vector specifies the damage $a_i \geq 0$ that the attacker will do to space station $i$, and must satisfy that $\sum_i a_i \leq 1$. When the attacker attacks with vector $a$, $a_i$ units of energy get subtracted from space station $i$, for each $i$.

(b) If as a result of the attack, any space station has negative (or zero) energy, it has been destroyed, and you lose the game.

(c) Otherwise, you get to pick one space station now, and repair it completely, restoring its energy all the way to $c$. 
Prove that if $c \leq H_n$ (where $H_n = \sum_{i=1}^{n} 1/i$ is the $n^{th}$ Harmonic number), then no deterministic online algorithm can survive forever against a perfect attacker.

(7) Chocolate Problems (1 chocolate bar each): For the previous problem, do the following two:

(a) Assuming that $c > \alpha \cdot H_n$ (where $\alpha$ is a constant of your choice; ideally, $\alpha = 1$, but larger $\alpha$ are acceptable), give a deterministic online algorithm, and prove that it will survive forever even against a perfect opponent.

(b) Prove that when the problem is not online, i.e., you get to know all the future attack vectors $a^{(t)}$ for all times $t$, then whenever $c > 2$, there is a strategy you can play to survive forever.