Due on April 25, 2018
30 points + 10 bonus points

General Directions: If you are asked to provide an algorithm, you should clearly define each step of the procedure, establish its correctness, and then analyze its overall running time. There is no need to write pseudo-code; an unambiguous description of your algorithm in plain text will suffice.

All the answers must be typed, preferably using LaTeX. If you are unfamiliar with LaTeX, you are strongly encouraged to learn it. However, answers typed in other text processing software and properly converted to a pdf file will also be accepted. Before submitting the pdf file, please make sure that it can be opened using any standard pdf reader (such as Acrobat Reader) and your entire answer is readable. **Handwritten answers or pdf files that cannot be opened will not be graded and will not receive any credit.**

Finally, please read the detailed collaboration policy given on the course website. You are **not** allowed to discuss homework problems in groups of more than 3 students. **Failure to adhere to these guidelines will be promptly reported to the relevant authority without exception.**

Problem 1 (20 points)
Consider the following decision problem: given an undirected graph $G = (V, E)$, a set of terminal vertices $T \subseteq V$, and a fixed parameter K, determine whether there exists a subgraph G' of G such G' connects all vertices in T (i.e., there exists a path between any $t_1, t_2 \in T$ in G') and includes at most K non-terminal vertices.

Prove that the above problem is NP-complete. Note that this involves showing:

(a) (5 points) The above problem is in NP.

(b) (15 points) The above problem is NP-hard.

(**Hint:** *You can use the fact that the vertex cover problem that we saw in class is NP-hard.*)

Problem 2 (10 points)
Let $G = (V, E)$ be an undirected graph. In the max-cut problem, we need to find the cut containing the maximum number of edges. Consider the following GreedyCut algorithm: start with an arbitrary cut, and repeatedly move any vertex to the other side if that increases the size of the cut.

(a) (5 points) Prove that the GreedyCut algorithm runs in polynomial time.

(b) (5 points) Prove that GreedyCut is a 2-approximation algorithm for the max-cut problem.

Problem 3 (10 bonus points)
This question is about your experience in the course. You will get full credit for answering this question, irrespective of the actual answer.

(a) (5 points) What did you like the most about this class?

(b) (5 points) Give one suggestion for improvement in future offerings of this class.