PRACTICE Exam 3

Name: _____

NOTES:

- This is a practice exam. It will not be graded. The aim is to give you a sense of the types of questions that may appear on the exam.
- As a reminder, this is not an exhaustive list of types of problems, anything through Tuesday November 5 is fair game.
- I recommend preparing your notes sheet before attempting these problems, then revising it to see if it needs to be adapted.
- Attempt all problems. On the exam, it is possible to receive partial credit for showing your work.

Problem 1	Score: /

Consider the following language:

$$L = \{ \langle R, S \rangle \mid R \text{ and } S \text{ are regular languages and } L(R) \subseteq L(S) \}.$$

Prove that L is decidable.

Problem 2	Score: /

Consider the problem of determining whether a two-tape Turing machine ever writes a nonblank symbol on its second tape when it is run on input w. Formulate this problem as a language, and show that it is undecidable with a proof by contradiction.

Problem 3	Score: /

Consider the following language $Poly$:

$$Poly = \{ \langle M, w \rangle \mid \text{Turing machine } M \text{ accepts input } w \text{ in polynomial time} \}.$$

Can we prove that this language is undecidable via Rice's Theorem? If so, use Rice's Theorem to show it is undecidable. If not, explain why in a few sentences.

Problem 4	Score: /

A *computer virus* is a computer program that replicates itself on a host machine and spreads to other machines, frequently by exploiting security vulnerabilities in said machines. A virus is frequently not benign; it typically performs some kind of negative behavior alongside replication, e.g., encrypting and holding users' files for ransom or stealing data or processing time as with a botnet.

Antivirus software has been developed to combat this threat, and has become an industry of its own. Top antivirus packages receive updates every couple of days so that they can better detect and defeat viruses currently in the wild.

The effort that goes into detecting and cleaning computer viruses is huge. It would be much better if there was some way for us to write a ”universal vaccination” program that would detect any possible kind of viral behavior and warn the user.

Use the ideas that we introduced regarding computational undecidability to argue why that such an endeavour—developing the ”perfect” antivirus software is practically impossible. Furthermore, describe what computational undecidability tells us we can expect from antivirus software insofar as accuracy goes.

Problem 5	Score: /

Consider that you are working at a social network company and you are tasked with the following problem:

Given a social network graph where the nodes are people and the edges are friend relationships, find largest collection of people that are all mutually friends with each other.

Recalling the three pillars of the intractability problem, choose one of the pillars to compromise on, and describe a high-level strategy to solve this problem with this compromise in mind. Describe the pros and cons of compromising on this pillar and the plausibility of this approach in real life.

(*Note*: I am interested less in whether you pick a ”correct” approach and more interested in seeing your rationale and justification based on our class discussions.)

Problem 6	Score: /

In banking, *float* is the time between making a payment with a check and the time that the funds for payment are deducted from your bank account. Before the advent of quick electronic check-clearing, it was advantageous for companies to open many bank accounts in disparate locations to maximize float. By doing so, these companies can maximize the interest on their accounts before their funds are deducted.

Imagine that we are such a company well on our way to maximize profits. Define the following sets of values:

- Let B be a set of banks with which we can open bank accounts.
- Let P be the set of people to whom we make payments.
- Let $v_{ij} \geq 0$ be the float created when we pay payee $j \in P$ from bank account $i \in B$.

Furthermore, suppose that we are limited to opening at most k bank accounts. Each bank will be used to pay one of our people although one bank can pay multiple people if it is optimal to do so.

Our goal is to find a set of banks $S \subseteq B$ with $|S| \leq k$ that maximizes our total float among all of payees. To do so, we observe that from our chosen subset S, for each payee j we want choose the bank i from S that maximizes v_{ij}. Thus, our overall objective is to find a subset S that maximizes the following function $V(S)$:

$$V(S) = \sum_{j \in P} \max_{i \in S} v_{ij}.$$

Give a natural greedy approximation algorithmfor this problem. You do not need to prove a bound for this algorithm, but you should argue (in a few sentences) why your choice of greedy heuristic is likely to be "optimal" insofar as a greedy approach might go.

(*Hint*: start with an empty set of banks. Until we reach the limit k, greedily choose a new bank to add to the solution. How can we use $V(S)$ to determine which bank to add at each step?)