Theory of Computations
Practice Final Exam Solutions

Name: _____

Directions:
Answer the questions as well as you can. Partial credit will be given, so show your work where appropriate. Try to be precise in your answers in order to maximize your points.

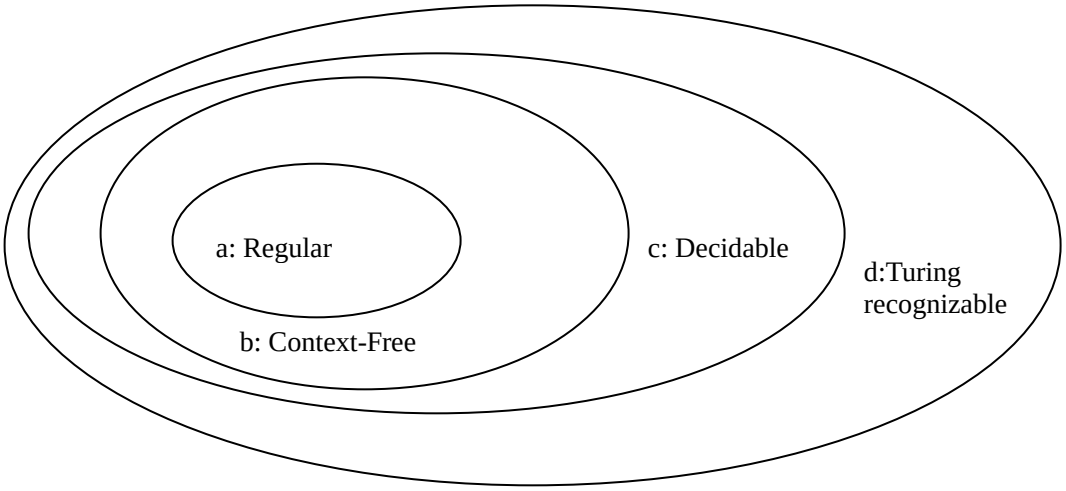
	Points	Score
1	**23**	
2	**14**	
3	**7**	
4	**7**	
5	**7**	
6	**10**	
7	**5**	
8	**5**	
9	**9**	
10	**13**	
Total	**100**	

1. **True/False questions**: For each part, circle either True or False. (23 points: 1 points each)

a.	A TM can compute anything a desktop PC can, although it might take more time.	[True]	False
b.	A Push-Down Automata can compute things that a TM cannot compute.	True	[False]
c.	Every Turing-decidable language is also Turing-recognizable.	[True]	False
d.	The Halting problem is decidable.	True	[False]
e.	All problems have an algorithm that will solve/decide them.	True	[False]
f.	$4n^2 = O(n)$	True	[False]
g.	$3n^3 = O(n^3)$	[True]	False
h.	$n\log n = o(n^2)$	[True]	False
i.	$2^n = O(n^{20})$	True	[False]
j.	You <u>cannot</u> build a DFA to recognize $\{0^{500}1^{10000} \cup 1^{1000}0^{200}\}$	True	[False]
k.	NP is the class of languages with polynomial time verifiers.	[True]	False
l.	The various sorting algorithms (e.g., bubblesort, heapsort) are in NP	[True]	False
m.	Most theoretical computer scientists believe that P = NP.	True	[False]
n.	All languages are Turing-recognizable	True	[False]
o.	The class of regular languages is closed under union	[True]	False
p.	All languages are decidable	True	[False]
q.	A regular language L may not be context-free.	True	[False]
r.	A_{DFA} is decidable. $A_{DFA} = \{<B,w> \mid B \text{ is a DFA that accepts input string } w\}$.	[True]	False
s.	Deterministic and non-deterministic PDA's have equivalent expressive power.	True	[False]
t.	If a problem A is reducible to problem B, then problem A must be no harder than B.	[True]	False
u.	An algorithm implemented on a single tape Turing machine will always have the same running time (e.g., big-O value) when run on a 2-tape Turing machine.	True	[False]
v.	NP is the class of languages decided by some nondeterministic polynomial time Turing Machine.	[True]	False
w.	From a computability perspective, every multi-tape Turing machine has an equivalent single-tape TM.	[True]	False

2. **Short answer questions.** Answer each question in a few sentences. (14 points: 2 each)

   a. The diagram below show a hierarchy of the languages we learned, with respect to computability. Write the proper language next to the labels a-d in the diagram below such that the hierarchy is correct. The languages are: Turing-recognizable, regular, decidable, context-free.



   b. A finite automata will run until its input is completely processed and then it will stop. This is not true for a Turing machine. Explain why.

   The input is placed on the tape of the Turing machine. The Turing machine runs until either the accept or reject state is entered—it is not like a finite automata which processes the input and then accepts if in an accept state or else it rejects. Thus, the TM does not have a notion of processing the input (whereas the FA does).

   c. A language is Turing-recognizable if some Turing machine recognizes it (this is a definition). But what does it mean when we say that a TM recognizes a language? The answer can be quite simple (one sentence) but please be precise.

   A TM recognizes a language if it <u>accepts</u> all elements in the language. That is all you need to say. It may not loop or reject or loop on any of these elements. You could also say (but you don't need to) that if an element is not part of the language, it can either reject or loop on it.

d. A language is Turing-decidable if some Turing machine decides it. What does it mean for a Turing machine to decide a language? Again, please be precise, but you can be relatively informal.

A TM decides a languages if it accepts all elements in the language and rejects all elements not in the language. Technically, it is not sufficient to just say that it halts on all inputs.

e. We are given a problem and find out that it is undecidable. Could there be an algorithm to solve it in polynomial time? Answer "yes" or "no" and then explain/justify your answer.

You can compute a solution to a problem only if you can always answer the problem. Thus, a problem must be decidable to be solvable. If it is not decidable, you cannot come up with an algorithm to solve (i.e., decide) it. You may be able to solve it in some cases, for example, if the language is Turing-recognizable you can recognize elements in the language, but typically when we say an algorithm solves a problem, we mean it solves it given all possible inputs.

f. I tell you that an algorithm runs in $O(2^n)$ but yet is in P. How can this be?

The big-O notation is not a tight bound. Thus any algorithm that runs in polynomial time (e.g., n^2) will also run in exponential time, such as $O(2^n)$. Thus, an algorithm that runs in $O(2^n)$ could be in P.

g. Assume that someone finds a polynomial time solution to an NP-complete problem. 1) What does that say about all NP-complete problems? 2) What does that say about the question of whether P = NP?

If someone finds a polynomial time solution to an NP-complete problem, then 1) all NP-complete problems can be solved in polynomial time and 2) P would then equal NP.

3. (7 points) Draw the DFA that, for the alphabet $\Sigma = \{0,1\}$ accepts all strings that do not have any consecutive 0's or 1's (e.g., 0, 0101, 101, and 10 are accepted but 11, 001, and 01001 are not)

4. (7 points) Provide a Context Free grammar that generates the language 00*1*.

   $S \rightarrow AB$

   $A \rightarrow 0A|0$

   $B \rightarrow 1B|\varepsilon$

5. (7 points) Let $E_{DFA} = \{<A>|A \text{ is a DFA and } L(A) = \varnothing\}$. Prove that E_{DFA} is decidable by providing an algorithm that decides it. You should also comment on why the provided algorithm is decidable.

   This is directly from the textbook (page 168) and we went over it in class.

   T = "On input <A> where A is a DFA:

   1. Mark the start state A

   2. Repeat until no new states get marked

   3. Mark any state that has a transition coming into it from any state that is already marked

   4. If no accept state is marked, accept; otherwise reject.

   This algorithm is decidable since it makes progress at each iteration (loop starting at line 2) since at least one node will be marked off and there is no possibility of an infinite loop.

6. (10 points total) Give implementation level descriptions of Turing machines that decide the following languages over the alphabet {a,b,c}. Recall that implementation level is lower level than the pseudo code that we use to describe algorithms (at the implementation level you talk about scanning the tape and tape movements).

   {w| w contains more than 5 times as many b's as a's}

   On input string w:

   1. *Repeat as long as an "a" remains on the tape:*
      2. *Scan the tape and mark the first "a" that has not been marked. Then return the tape head to the start of the tape.*
      3. *Scan the tape and mark the first 5 b's that are found and then return the tape head to the start of the tape. If you fail to find any of these b's, then reject.*
   4. *Scan the tape until you reach a first unmarked "b". If you reach it, then accept and if you do not, then reject (since there are exactly 5 times as many b's as a's).*

7. (5 points) Let A_{DFA} = {<B,w>| B is a DFA that accepts input string w}. Provide a simple explanation (one could call it a very informal proof) for why this language is decidable.

   We know how to compute with a DFA. Simply feed the input string into it and follow the transitions specified in the DFA. As we compute, we always get closer to termination, since the DFA will terminate when the input string is done being processed. Thus, since we make progress every step and there is no way to loop, we are guaranteed to halt. If we halt in the accept state, then accept; else reject. So, this language is decidable since the DFA will always halt and give an answer.

8. (9 points) Describe a PDA that accepts the following languages. For part b, if you want, rather than describing it from scratch, you can just say how you would modify your answer from part a. Note: part b is a bit tricky. Hint: for part b, you will need to utilize a capability of the PDA not used in part a.

   a. $L = \{0^m1^n: n \leq m\}$

   Start in a state that will take any leading 0's and push them onto the stack.

   When you see a 1, transition to a new state and start popping a 0 from the stack for each 1 that is seen. If any subsequent 0's are seen, immediately reject. If you run out of 0's to pop from the stack, then reject (this means n > m). When you complete processing the input, accept (you would have rejected before this point if the input did not belong to the language).