CS 3120 Final Cumulative Quiz

This is this final cumulative quiz for DMT2. This **cover sheet** is here to provide instructions, and to cover the questions until the quiz begins. **do not remove this cover sheet** until your proctor instructs you to do so.

You will have the entire exam period to complete this quiz. There are three pages worth of questions. Make sure to **write your name and computing id at the top of the quiz**.

When you are done, you will come to the front of the room and turn in the quizzes (do NOT cut off the staple on this one!). The proctors will be available at the front of the room to clarify this if you have any questions.

This quiz is CLOSED text book, closed-notes, closed-calculator, closed-cell phone, closed-computer, closed-neighbor, etc. Questions are worth different amounts, so be sure to look over all the questions and plan your time accordingly. Please sign the honor pledge below.

The Tao that is seen
Is not the true Tao,
until You bring fresh toner.

Quiz - Final Cumulative Quiz

Name ______________________________

1. [16 points] Answer the following True/False questions.

We proved the *halting problem* is undecidable by showing that a decider for it could be integrated into a solution for A_{TM}	**True**	**False**
A *Turing machine* can be used to decide all *regular languages* and all *context-free languages*	**True**	**False**
Non-determinism increases the power of *DFA*s and *PDA*s, but not *Turing machines*	**True**	**False**
If a *polynomial time* algorithm for *Vertex Cover* is discovered, then we still won't know how to solve *SAT* efficiently because the reductions we did are in the wrong direction (from *SAT*, through other problems, to eventually *Vertex Cover*)	**True**	**False**
Every *PDA* that contains an infinite loop (e.g., and epsilon transition that triggers continuously) has an equivalent *PDA* that halts	**True**	**False**
Every *Turing machine* that contains an infinite loop has an equivalent *Turing machine* that halts	**True**	**False**
For any two problems $A, B \in P$, $A \leq_p B$. But it is not necessarily the case that a faster reduction exists (e.g., $A \leq_{n^2} B$ may not exist)	**True**	**False**
Verification Problems will never be worse than *polynomial time* because verifying a solution is always easy	**True**	**False**
When converting an *NFA* with n states into a *DFA*, the *DFA* that is produced may have $O(2^n)$ states	**True**	**False**
A *Turing machine* with any number of tapes (e.g., 6 tapes) will still be as powerful as a traditiona single-tape *Turing machine*	**True**	**False**
All *DFA*s and *NFA*s recognize only problems in P	**True**	**False**
All *PDA*s recognize only problems in P	**True**	**False**
A *Turing machine* can *recognize* all *NP-Complete* problems	**True**	**False**
A *Turing machine* can *decide* all *NP-Complete* problems	**True**	**False**
The substitutions that generate a string with a specific *Context-free grammar* can be simulated by using a stack	**True**	**False**
An *odd* number of these true/false questions are *true*	**True**	**False**

2. [4 points] For each machine type we have seen and proposed alteration to that machine, select whether it causes the set of recognizable functions to decrease, increase, or remain unchanged.

Machine	**Alteration**			
DFA	Can be in 1, 2, or 3 states at once	**Decrease**	**Increase**	**Unchanged**
NFA	Has access to a *stack* with finite size	**Decrease**	**Increase**	**Unchanged**
PDA	Has access to a *queue* in addition to the *stack*	**Decrease**	**Increase**	**Unchanged**
DTM	Head of machine can only move right (not left)	**Decrease**	**Increase**	**Unchanged**

3. [6 points] For each language below, choose whether it is regular, context-free, turing-decidable, turing-recognizable, or none (not even turing recognizable).

w where w is binary representation of a prime number	**Reg**	**Cont. Free**	**Tur. Dec.**	**Tur. Rec.**	**Not Tur. Rec.**
w where w does NOT contain three 0s	**Reg**	**Cont. Free**	**Tur. Dec.**	**Tur. Rec.**	**Not Tur. Rec.**
w where w contains the same number of 0s and 1s	**Reg**	**Cont. Free**	**Tur. Dec.**	**Tur. Rec.**	**Not Tur. Rec.**
M, w where Machine M writes a 1 to tape cell 6 during computation	**Reg**	**Cont. Free**	**Tur. Dec.**	**Tur. Rec.**	**Not Tur. Rec.**
M, w where Machine M NEVER writes a 1 to tape cell 6 during computation	**Reg**	**Cont. Free**	**Tur. Dec.**	**Tur. Rec.**	**Not Tur. Rec.**
0^n0^n100	**Reg**	**Cont. Free**	**Tur. Dec.**	**Tur. Rec.**	**Not Tur. Rec.**

In class, we stated (but did not prove) that *finite automata* could be simulated using a *Turing machine*. For this question, you will provide some of the details for exactly how a generic *NFA* could be converted into an equivalent *single-tape Turing machine*. You can use any proof from class without regurgitating the details, but you will likely need to add some detail in addition to what was seen in class.

4. [2 points] First, describe your overall, high level, approach to doing this conversion. Where convenient, you can invoke proofs / conversions from class and even chain them together if it is helpful.

5. [2 points] **Detail 1**: Describe in 1-2 sentences how your approach removes *non-determinism* from the original NFA.

6. [2 points] **Detail 2**: Describe in 1-2 sentences how your approach handles the difference between how *NFA*s and *DTM*s treat input.

7. [2 points] **Detail 3**: Describe in 1-2 sentences how your approach keeps track of the state of the original *NFA*.