Fall 2024 COT 4210 Final Exam
Date: December 3, 2024

First Name: _____ Last Name: _____

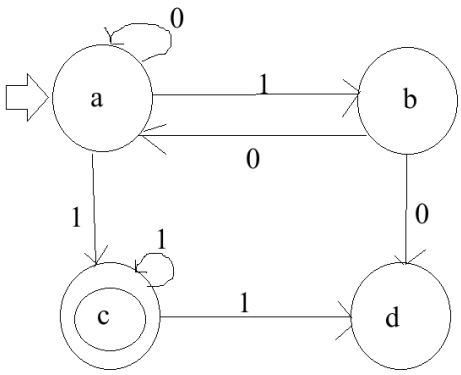
1) (8 pts) Design a DFA over the language {0, 1} for all strings which contain the same number of occurrences of the substring 01 and 10. (Note, ϵ, 0 and 1 are in the language since each of these strings has 0 occurrences of 01 and 0 occurrences of 10.)

2) (8 pts) Please give the formal specification (no drawing) of your DFA from question 1. As long as your answer for #1 is a valid DFA of roughly the same complexity as the correct answer and your description here matches your drawing for question 1, you'll get full credit.

3) (10 pts) Use the pumping lemma for regular languages to prove that the set of strings over the alphabet {0, 1} with the same number of occurrences of the substrings 00 and 11 is not regular.

4) (4 pts) Succinctly explain why picking the string $s = (0011)^p$ would have rendered completing the proof for number 3 impossible.

5) (10 pts) Convert the NFA drawn below to an equivalent DFA. Please label the states of your DFA as subsets of {a,b,c,d} to aid the graders and draw a box around your final DFA. Clearly indicate the start state, accept state and ALL transitions. Draw a box around your final DFA.



6) (10 pts) Below is a grammar over the alphabet Σ = { ‘(‘, ‘)’, ‘+’, ‘x’, ‘a’, ‘b’} where the first two items are parentheses, the next two items are operators (addition, multiplication) and the last two items are operands (representing numbers).

S → (E) + (E) | (E) x (E)
E → (E) + T | (E) x T | T
T→ a | b

(a) Show the derivation of a string that is part of the language described by this grammar that is equivalent algebraically to $a^2b + a(a + b)$. You may make multiple substitutions in a single step so long as it’s clear which individual rules in which locations you are using.

(b) Draw a possible parse tree for the derivation in part (a).

7) (10 pts) We define a dead state, s, of a Turing Machine, M, to be a state that M NEVER enters, no matter what input is given to M.

Let L = { <M> | M is a Turing Machine, that has a dead state.}

Prove that L is undecidable.

8) (15 pts) The set of Lattice points on the Cartesian plane is countable. (This is all ordered pairs (x, y), where x and y are both integers. We can specify an order of these points to prove their countability. First, use the following ordering of the integers: 0, 1, -1, 2, -2, 3, -3, etc. Next, create a table with the integers in the order specified as both row and column labels. (Our rows are row 0, row 1, row -1, etc. Same for our column labels.) To create a one-to-one mapping with the positive integers, we can always agree to start at the beginning of each row and read up the forward diagonal, moving to the next row once we reach the first row. (Row label is for x, column label is for y.) Following these rules, the first seven ordered pairs in our ordering would be (0, 0), (1, 0), (0, 1), (-1, 0), (1, 1), (0, -1), and (2, 0). Complete the method below, so that if it's given an x-coordinate and a y-coordinate as input, it returns the **1-based** rank of that ordered pair in this ranking system. (For example, if the input is x=0, y=0, the output should be 1.) Do not worry about overflow issues. For full credit, your code must run in O(1) time. You may write a helper method if you wish. (If you do, please do so below the end of the getRank method. **CODE MUST BE IN JAVA!!!**

```
public static long getRank(long x, long y) {

}
```

9) (10 pts) Write a O(nlgn) time verifier, where n is the size of the list S, for the PARTITION language. PARTITION = {S | S is a list of numbers such that there exists a way to split S into two groups where the sum of the numbers in the two groups is equal}. The verifier should take in two arrays: **`values`**, representing all of the numbers in the set S for input and **`part1`**, storing the 0-based indexes into values representing which numbers to place in the first group for the partition. Your verifier should return true if the certificate stored in **`part1`** represents a valid partition of **`values`** and false otherwise. (For example, if **`values = {2, 6, 3, 1}`** and **`part1 = {1},`** then your function should return true, since {6} (part1) and {2, 3, 1} (other group) have the same sum of values. (Note: The length of an array named array in Java is array.length.) **You must check that part1 has unique integer values that are all valid indexes into values. If any index is repeated or any index is out of bounds, your method should return false.** (Note: Arrays.sort(a) sorts the array a.)

```
public static boolean isPartition(int[] values, int[] part1) {

}
```

10) (10 pts) In question 9 you essentially proved that the PARTITION language is in NP. For this question, you'll prove it's NP-Complete by doing a mapping reduction from SUBSET-SUM to PARTITION. (Note: This is harder that doing the mapping reduction from PARTITION to SUBSET-SUM and no credit will be given for a reduction in this direction.) **For the purposes of this problem let's assume the only valid input numbers for the lists in both the SUBSET-SUM and PARTITION problems are non-negative integers.**

11) (5 pts) Panama City is the capital of which North American country? _____