Theory of Computation, CSCI 438, Spring 2021
Exam 3, ~~March 26~~ March 31st

Name _____

Essay portion of the exam. (10 pts.)

Definition

1. Define what it means for a language to be Turing-recognizable (also called recursively enumerable). (5 pts.)

   Definition 3.5 (page 170): A language is Turing-recognizable if some Turing machine recognizes it.

Short Answer

2. Describe the difference between a language being Turing-recognizable and Turing-decidable. (5 pts.)

   Turing-decidable languages always halt. Turing-recognizers may or may not halt.

   Both TMs accept all strings in the language. The difference lies in what the TM does when the string is not in the language. Deciders reject. Recognizers either reject or loop forever.

3. Give the inputs to each of the following languages: (10 pts.)

   $E_{RegExpression}$ A single regular expression

   $EQ_{RegExpression}$ 2 regular expressions

   $A_{RegExpression}$ A regular expression and a string

Problem Solving

4. Let ALL_{DFA} = {<A> | A is a DFA and $\mathcal{L}$ (A) = Σ^*}.
   Show that ALL_{DFA} is decidable. (10 pts.)

   D = "On input <M> that encodes a DFA
   i. Translate M to a machine that accepts the complement of what M accepted (i.e. make non-accepting states accepting, and vice versa)
   ii. Run E_{DFA} on the new machine.
   iii. If E_{DFA} accepts the complement of M, accept; If E_{DFA} rejects the complement of M, reject"

5. Fill out the columns, telling if the language is regular, context-free, decidable and/or Turing recognizable and how you would show this. (15 pts.)

Needs fixing

Language	The language is regular	The language is context free	Decidable	Turing recognizable
L={w | w is a palindrome}	No, pumping lemma for regular languages	Yes, create a grammar or a pda for it	Yes, create a Turing machine that always halts, or say that all context-free languages are also decidable	All decidable languages are Turing recognizable
L={w such that w begins and ends with the same symbol }	Yes, create a DFA, NFA or regular expression for it	Yes, create a grammar or it or state that all regular languages are context free	Yes, create a Turing machine that always halts	All decidable languages are Turing recognizable
L={$a^n b^m c^k$ | n=m or m=k}	No, pumping lemma for regular languages	Yes, create a grammar or a PDA for it	Yes, create a Turing machine that always halts	All decidable languages are Turing recognizable

6. Create a Turing machine that recognizes
   L = {w | w∈{0,1}* and w contains twice as many 0s as 1s}.

High-level plan: (10 pts.)

Mark the front of the tape.
Repeatedly, for each 1, mark off two 0s. Rather than be efficient, always start at the front of the tape.

Detailed plan: (10 pts.)

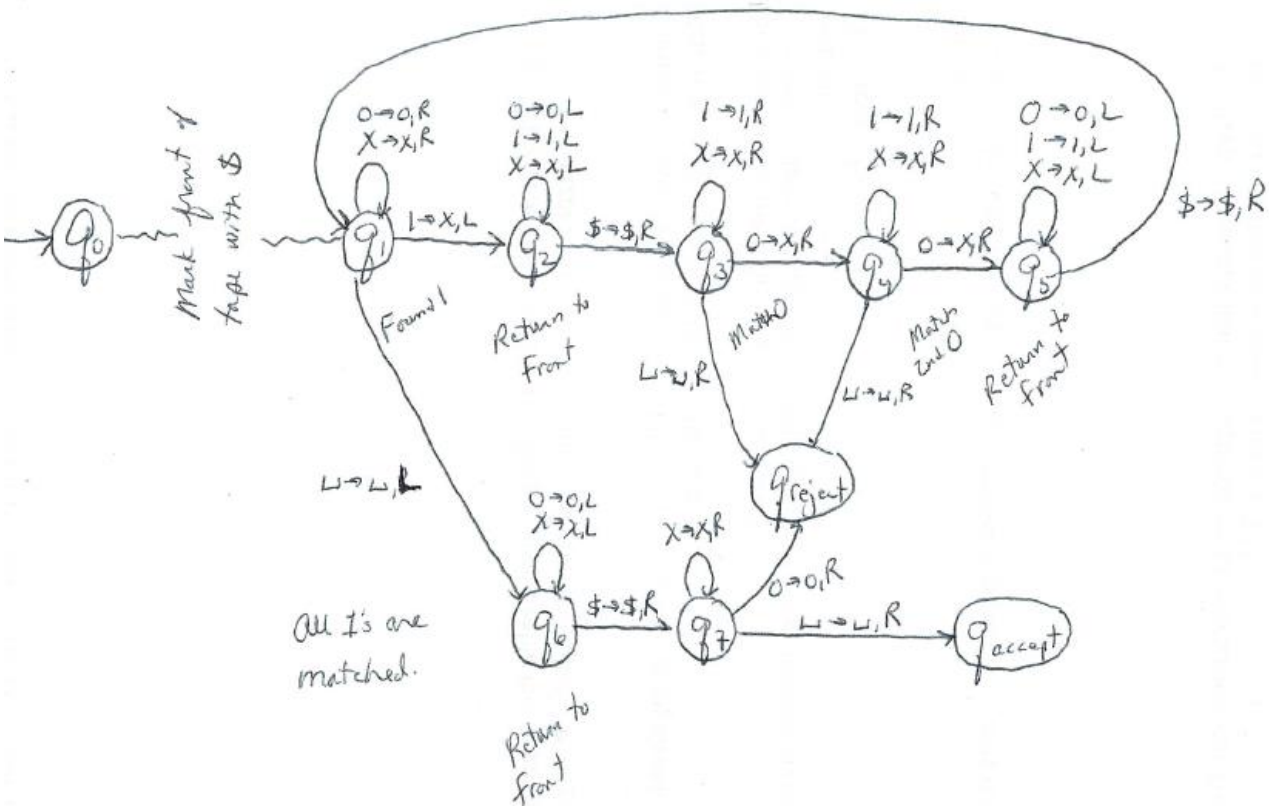
Mark the front of the tape with $, shifting all non-blank symbols one position to the right.

loop {
    travel right over '0's and 'x's until see '1'. Replace '1' with 'x'.
        if '_' exit loop
    return to the front of the tape
    travel right over 'x's and '1's until see '0'. Replace '0' with 'x'.
        if '_', reject
    continue traveling right over 'x's and '1's until another '0'.
    Replace '0' with 'x'.
        if '_', reject
    return to the front of the tape
}

// Matched all '1's with two '0's.
return to the front of the tape
travel right over 'x's
    if see '_', accept.
    if see'0', reject

Turing machine: (5 pts.)

Possible answer:



Proof

7. Let ADD be the language defined by
ADD={a=b+c | a, b, c are binary integers, and a is the sum of b and c}.

The alphabet for ADD is {0, 1, +, =}. Binary integers are any non-empty combination of 0s and 1s.

If ADD is context-free, create a CFG or a PDA for it. If it is not context-free, prove that it is not using a pumping lemma. (20 pts.)

Proof: Suppose, by way of contradiction, that ADD is context-free. Then the pumping lemma must hold for ADD. Let p be the pumping length. Consider the string
s= "$1^p0^{p+1}=1^p0^p+1^p0^p$". Clearly $s \in$ ADD and $|s| \geq p$. The pumping lemma guarantees that s can be divided into five pieces s=uvxyz where $|vxy| \leq p$, $|vy|>0$ and $s_i=uv^ixy^iz \in$ ADD for all $i \geq 0$. Consider all possible breakdowns s=uvxyz where $|vxy| \leq p$, $|vy|>0$.

Case 1: vxy occurs entirely to the left of the "=".
Case 2: vxy contains the "="
Case 3: vxy occurs entirely in the 1^p0^p between the "=" and "+"
Case 4: vxy contains the "+"
Case 5: vxy occurs to the right of the "+".

Consider each case separately.

Case 1: vxy occurs entirely to the left of the "=".
In this case s_0 will be syntactically correct, but the arithmetic is wrong since the right side sums to 1^p0^{p+1}, but the left side is something less than 1^p0^{p+1}. Thus, for each decomposition of Case 1, s_0 is not in ADD. Thus, for each decomposition of Case 1, this is the not decomposition which is pumpable.

Case 2: vxy contains the "="
If either v or y contain all the "=", s_2 will contain two equal signs, and the statement is syntactically incorrect.

Say that x contains "=". Then the string s_0 will be syntactically correct, but the arithmetic will be wrong because there is no way to remove 0s from the first binary digit, and 1's from the second binary digit, and make the sum work. Thus, s_0 is not in ADD.

Thus, for each decomposition of Case 2, the string is not pumpable.

Case 3: vxy occurs entirely in the 1^p0^p between the "=" and "+"

In this case s_0 will be syntactically correct, but the arithmetic is wrong since the left side of the equation is 1^p0^{p+1}, but the right side sums to something less than 1^p0^{p+1}. Thus, for each decomposition of Case 3, s_0 is not in ADD. Thus, for each decomposition of Case 1, this is the not decomposition which is pumpable.

Case 4: vxy contains the "+"
If either v or y contain all the "+", s_2 will contain two plus signs, and the statement is syntactically incorrect.

Say that x contains "+". Then the string s_0 will be syntactically correct, but the arithmetic will be wrong because the left side of the equation is 1^p0^{p+1}, but the right side sums to something less than 1^p0^{p+1}.

Thus, for each decomposition of Case 4, the string is not pumpable.

Case 5 is similar to Case 3.

Since Cases 1-5 cover all possible decompositions, and it has been shown that none of these hold a pumpable decomposition, it must be concluded that the pumping lemma does not hold for ADD. Thus ADD must not have been context-free.