Name: Arpit [redacted]

Student ID: [redacted]

CS181 Winter 2018 - Midterm

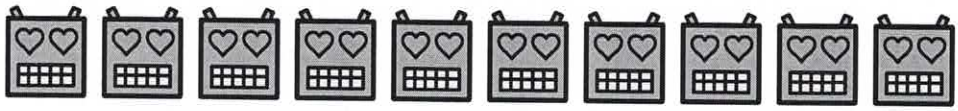
You will have 110 minutes to take this exam. This exam is open-book and open-notes, but any material not used in this course is prohibited. There are two questions and a total of 150 points and the second question has one part worth 30 extra credit points. You will **not** get 20% points for just writing "I don't know". However, if you describe a non-trivial approach that you tried using to solve the problem but realized it doesn't work and explain correctly why it doesn't work and then write "I don't know" you will get 20% points for that problem. It is solely the grader's discretion as to whether your stated approach was indeed non-trivial. Place your name and UID on every page of your solutions. **Please use separate pages for each question.**

Honor Code Agreement: I understand this exam is open-book and open-notes, but any material not used in this course is strictly prohibited. I also understand that this exam is to be taken individually without any outside help (except possibly from the professor or the TAs) within the time limits set forth. I agree to adhere to the course honor code and if I am unsure of any rules of the honor code, I will ask for clarification from the professor or the TAs.

Signature: Arpit

Question	Points
1	55
2	79
EC	0
Total	134

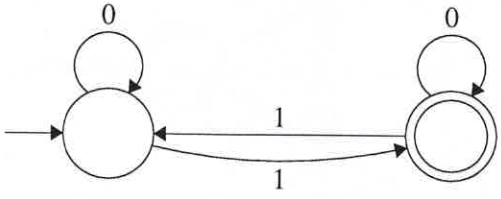
Note: While solving the problems, you can always reference theorems and claims from your homeworks, lectures, and anything in the book without having to prove those again.



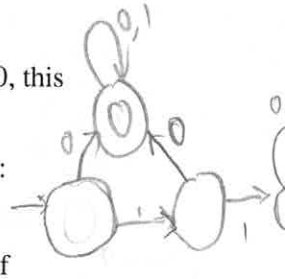
Problem 1. A *Lovesick Robot Automaton (LRA)* is a new type of machine which generates an infinite pattern. We think of the LRA as starting at position 1 and moving right in a sequence of steps, and at each step it decides to either mark the current position with a ♡ or to not mark anything, i.e. it leaves a blank character, which we denote with "␣". The LRA has the following characteristics:

- Just as in the case of a DFA, a LRA is a five tuple $(Q, \Sigma, \delta, q_0, F)$. However, after a LRA has processed the last character of the input string, it goes back to the first character. So a LRA processes the string repeatedly in an infinite loop. Each step of the robot corresponds to a state transition.
- The set F is now called the set of *marking states.* At each step, if the new state is a marking state then the robot marks the new position with a ♡. Otherwise, the robot does not make a mark in this position, i.e. it leaves it blank (␣)
- It is only after processing the first symbol of the input that the LRA either creates a mark ♡ or leaves a blank ␣, depending on what state the LRA enters after processing this first symbol.

Example: Consider the following LRA, given by the diagram.



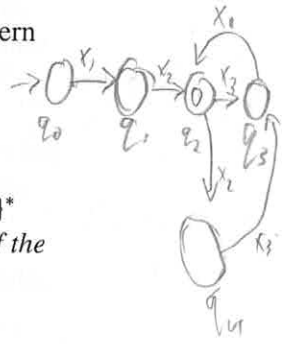
On the string 000, this robot generates the pattern ␣␣␣␣␣␣␣... and on the string 10, this robot generates the pattern ♡♡␣␣♡♡␣␣♡♡␣␣♡♡␣␣...



a) **(15 points).** Describe a LRA with at most 5 states that has the following property:

- If the string starts with "11", the robot makes no marks.
- If the string starts with anything else, the robot makes an infinite sequence of marks.

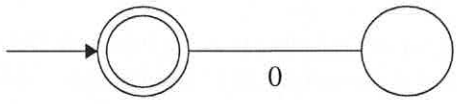
b) **(20 points).** Assume we have a LRA M and a string $x = x_1x_2x_3$ of length three, and the only thing we know about M and $x = x_1x_2x_3$ is the following: $M(x)$ enters the following sequence of states in the beginning: $q_0, q_1, q_2, q_3, q_2, q_4, q_3$, and the pattern starts out with ␣♡␣♡␣␣. Describe what the pattern will look like from this point onward, and informally describe why.



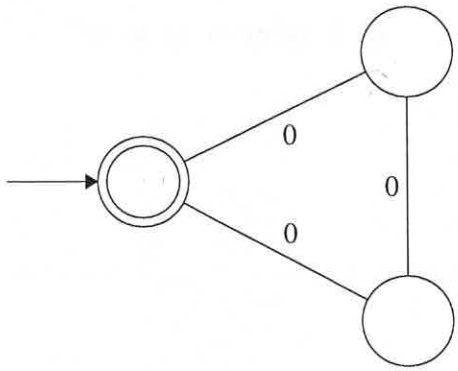
c) **(35 points).** Rigorously prove the following "pumping lemma" for LRAs:

Lemma 1. *For all LRAs M and strings s, there exists some $n \in \mathbb{N}$ and $w, z \in \{♡, ␣\}^*$ such that $|w| \leq n$ and the pattern generated by the robot when executing $M(s)$ is of the form $wzzzzz\ldots$.*

Problem 2. Loving relationships should be symmetric.
In that spirit, we study a variant of an NFA called a *Symmetric NFA (SNFA)*, which is just like an NFA except that for every $q, q' \in Q$ and $c \in \Sigma_\epsilon$, if $q' \in \delta(q, c)$ then we have that $q \in \delta(q', c)$. That is, if there is a transition from q to q' on character c then there is also one in the reverse direction on c. We draw this using an NFA diagram where all edges are undirected.
We say that a language is *bidirectional regular* if there exists a SNFA recognizing it.
Example: The language generated by $(00)^*$ is bidirectional regular, recognized by the following SNFA.



a) **(15 points).** Consider the following SNFA.



Describe the language that this SNFA decides.

b) **(15 points).** Assume we have a bidirectional language L and all we know about this language is that $0010 \in L$. Show that 00100110 is also in L.

c) **(30 points).** Formulate a lemma that generalizes the observation in (c), of the following form:

Lemma 2. *Let L be a bidirectional regular language. Then for all $w \in L, |w| > 0$, we can decompose w in the following way ⟨describe some way to decompose w⟩ such that ⟨use the decomposition you described to generate infinitely many strings in L⟩.*

Prove the lemma that you formulated. (Note that you cannot simply state and prove the pumping lemma for regular languages, since your lemma must apply to all nonempty strings $w \in L$, not just strings of a certain minimum length.)

d) **(20 points).** Give an example of a language that is regular but not bidirectional regular. Prove that this is the case.

e) **(30 points extra credit).** Consider a new type of machine which is exactly like a SNFA except that now it has a stack. That is, it is a PDA where for all $q, q' \in Q, c \in \Sigma_\epsilon, a \in \Gamma_\epsilon$ and $a' \in \Gamma$, if $(q', a', \text{push}) \in \delta(q, c, a)$ then $(q, a', \text{push}) \in \delta(q', c, a)$. Similarly, if $(q', a', \text{pop}) \in \delta(q, c, a)$ then $(q, a', \text{pop}) \in \delta(q', c, a)$ ***You should assume that the stack starts out with the "empty stack" character $. So you don't have to add that character manually.***

Prove or disprove the following statement: for any regular language L, there is a bidirectional pushdown automaton which decides L. Partial credit will be handed out more harshly for this problem.

Hint: Recall that a PDA does ***not*** *need to empty its stack in order to accept. The same applies to a SPDA. Also remember that partial credit is given out much more rarely for extra credit, so do not waste too much time writing a solution for this problem unless you're sure you have a solution that works.*

♥ ♥ ♥ ♥ ♥ ♥ ♥ ♥