CSE 311: Foundations of Computing I Spring 2017

Homework 1 (due Friday, March 31)

Directions: *Write up carefully argued solutions to the following problems.*

To simplify submission, we ask that you make sure only one *question is on each page. You may, of course, use multiple pages for one question if necessary.*

Additionally, we will be offering a small amount of extra credit for each homework that you typeset using LaTeX. We will hold a session on how to use LaTeX on Wednesday from 6:30pm to 8:30pm in EEB 105 to help get you started.

0. The Tortoise, the Hare, and Floyd [Written] (24 points)

For each of the first three parts, translate the English statement into propositional logic. Be sure to define exactly what your atomic propositions mean.

(a) [6 Points] If the tortoise walks at a rate of one node per step, and the hare walks at a rate of two nodes per step, then the distance between them increases by one node per step.

(b) [5 Points] I can create an algorithm that detects if a linked list has a cycle if I know this idea.

(c) [7 Points] Either the tortoise and the hare will eventually be on the same node, which implies the linked list has a cycle, or an animal will reach the end of the list (which implies the linked list does not have a cycle).

In this part, we will give you a bad translation and ask you to explain why it doesn't work.

(d) [6 Points] Consider the following sentence:

> My code is wrong, because I didn't debug it.

If we let p be "my code is wrong", and we let q mean "I didn't debug it", a tempting translation for this sentence is $q \to p$. Explain (in no more than two sentences) why this translation isn't reasonable.

1. Nanananananananagate! [Written] (16 points)

For each of the following parts, translate the English statement into logical language. Then, negate the statement using formal logic. Once you have negated the statement provide an easy to read English rendition of the sentence. Make sure to show your work along the way!

All formal logic sentences you provide should have any negation symbols immediately in front of propositional variables. For instance, $\neg(p \to q)$ is not simplified enough.

Your english sentences should avoid propositions like "not (Roger is orange)" which do not sound natural.

(a) [8 Points] If x and y are the same, then x must be zero.

(b) [8 Points] This course is the best one ever if and only if you are a student taking CSE 311.

Note: For this part, your final answer may not use any negation symbols at all.

2. The Curious Case of The Lying TAs [Online] (20 points)

A new UW CSE student wandered around the Paul Allen Center on their first day in the major. They found (as many do) that there is a secret room in its basements. On the door of this secret room is a sign that says:

```
All ye who enter, beware!  Every inhabitant of this room is either a TA who
always lies or a student who always tells the truth!
```

In this problem, you will model several situations in this peculiar version of the Paul Allen Center basement by writing Java programs. The point here is to draw a connection between the programming you've already done and what we're learning in CSE 311.

You should begin by going to `https://gitlab.cs.washington.edu/cse311-17sp/hw1-lying-tas/` and downloading the necessary source code for this problem.

Situation1: The CSE student walked into the room, and two inhabitants walked up to the student. One of them said "at least one of us is a TA."

Situation2: Three inhabitants walk up to the CSE student and surround the UW CSE student. One of them says "every TA in this circle has a TA to his immediate right."

When we consider different situations, you should *not* make the assumption that they both actually happened.

For each of the following parts, you will write code to determine which models of the situation are possible. You may only use a small subset of Java in your `sat` and `implies` functions, because we're trying to *only use propositional logic*. In particular. you may not use loops, if statements, data structures (boolean variables are okay though), or Strings.

(a) [5 Points] We have provided you with starter code for modeling **Situation1** with *two* inhabitants. To finish the model, you will need to fill in the `sat` and `implies` methods. We strongly recommend you read the comments in the `Situation1TwoPeople.java` file before beginning.

(b) [7 Points] Now that you have a correct implementation of **Situation1** with *two* inhabitants, we want you to generalize it. In particular, consider the same **Situation1**, but with the twist that there are now *three* inhabitants. Starting from your code in `Situation1TwoPeople.java`, create a new implementation called `Situation1ThreePeople.java` which models this slightly changed situation.

N.B.: It should feel like something is "wrong" as you generalize the problem this way. . . which feels weird. It's worth considering why this was so annoying. We will revisit this soon.

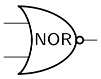
(c) [8 Points] Finally, now that you have a program to solve `Situation1ThreePeople.java`, you should solve **Situation2** in a file called `Situation2ThreePeople.java`.

N.B.: This time, it should feel natural and easy to make the changes. Again, it's worth thinking about why this is the case.

You can submit and check your answers to this question using
`https://grinch.cs.washington.edu/cse311/lying-tas`.

3. All You Need is Nor, Nor, Nor. . . [Online] (20 points)

The **NOR** connective takes two propositions and evaluates to True when both propositions are False and evaluates to False otherwise. In circuit diagrams, the gate for **NOR** is denoted by

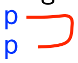


The **NOR** of p and q is written as $p \downarrow q$.

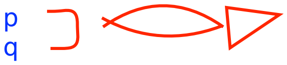
Demonstrate that we can construct *all the other connectives* by just using **NOR** by writing propositional formulae for each of the following while *only using **NOR** connectives*. You may *not* hard-code T or F in your circuit.

Hint: It's okay to use a single input/output more than once.

(a) [5 Points] $\neg p$

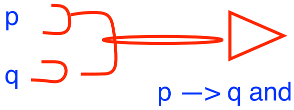


(b) [5 Points] $p \vee q$



if p and q are false, then first gate is true leading to false second gate.

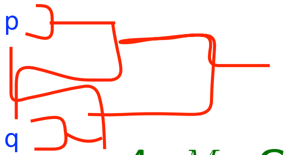
(c) [5 Points] $p \wedge q$



if not p and not q are both false, the p^q is true

(d) [5 Points] $p \leftrightarrow q$

p —> q and q —>p : not(p and not q) and not(q and not p): if p and not q and q and not p are both false then this is true. p and not q is true iff not p is false and q is false

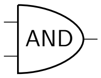


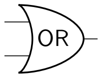
You can submit and check your answers to this question using https://grinch.cs.washington.edu/cse311/propositions.

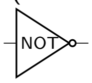
4. $\mathcal{M}$o Gates, $\mathcal{M}$o Problems! [Written] (10 points)

Using only. . .

AND Gates, **OR** Gates, and Inverters (**NOT** Gates),



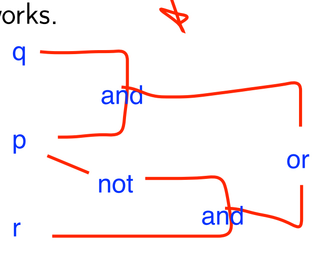




draw the diagram of a circuit with **three** inputs that computes the function $M(p, q, r)$, where the following defines M:

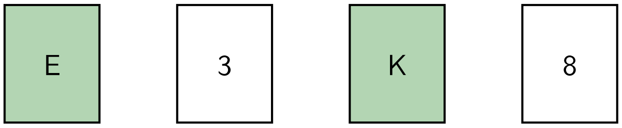
$$M(p,q,r) = \begin{cases} q & \text{if } p = \text{T} \\ r & \text{if } p = \text{F} \end{cases}$$

You may *not* hard-code T or F in your circuit. Explain why (or how) your answer works.



5. Getting Carded [Written] (10 points)

You are presented with four *two-sided* (one green, one white) cards:



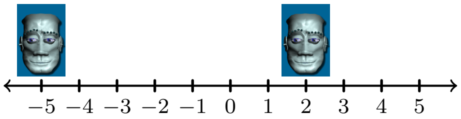
On the green side of each card is a letter, and on the white side is a number.
Consider the following rule:

> If a card has a vowel on one side, then it has an even number on the other side.

Which cards would need to be turned over to check if the rule is true? Explain your answer in a few sentences.

E and 3.
We have to verify that E has a vowel on the other side, and that 3 does not have a vowel (if 3 had a vowel, then it wouldnt be true that vowel —> even number)

6. EXTRA CREDIT: Robot Rumble (-NoValue- points)



Two robots are pushed out of an airplane onto one of the numbers on an infinite number line. Each robot leaves its parachute where it lands and starts executing its program. Your task is to write a program to be given to both robots that will always make them collide. Your program will be a string using only the following four commands:

- L: go left one unit
- R: go right one unit
- S: skip the next command, unless I'm on a parachute
- 0-9: Go to the nth command

faster: LS40LLLLL4
shorter code: LS40L4

Some more details:

- The program starts running from the 0th character.
- Your program may be at most 10 characters long.
- The robots each run one command at a time, simultaneously.
- Each command takes the same amount of time to run.
- The capitalization of the commands is important.

6LLLL1LS16
4LL1LS14

To submit this question, do the following two things:

(1) Submit your *answer* using the check script at: `http://www.countablethoughts.com/perfectly/symmetrical/violence/never/solves/anything/now.now.py`

(2) In your normal homework write-up, explain in a couple sentences how your answer works. For the TAs' sanity, please repeat your answer in your description.