EECS 649: PROBLEM SET 1

Reading:

- R&N 1.1-5, 27.1-3, Turing's "Computing Machinery and Intelligence" (linked)
- Some of these problems are from previous editions of R&N and other books, so use my problem numbering and disregard any numbers that may appear in problem scans.

Total Points: 100

Format:

- Use standard sheets of paper (8.5 by 11 inches)
- Perform all work neatly. When asked to write a paragraph (as in 1.1-4 below),they should be typed (e.g., using a computer and LaTeX or Word). Other work may bescanned to pdf for submission to Gradesope.
- Submit only a single (combined) PDF file to Gradescope.

Problems 1.1-4 are taken from the linked page above. Write a paragraph for each. **Answers for 1.1-4 must be typed.** A sample problem/answer/rubric similar to 1.1-4 appears below.

SAMPLE Problem (R&N)
Every year the Loebner Prize is awarded to the program that comes closest to passing a version of the Turing test. Research and report on the latest winner of the Loebner prize. What techniques does it use? How does it advance the state of the art in AI?

SAMPLE Answer (due to former student Nathan Makowski)

1.3 The most recent Loebner prize was Robert Medeksza who won for the Ultra Hal Assistant, a digital secretary that learns through interaction and remembers information that the user wants it to save.
Ultra Hal uses language processing in order to have natural conversations with the user. Ultra Hal advances the state of the art by keeping information the user inputs, learning voice recognition and learning about the user. It also interacts with other computer programs in order to store information as well as to help research different topics or access the web.

SAMPLE Grading Rubric
Student has substantially and correctly answered the question (8-10 points). Points will be taken off for missing a piece of the question (e.g., how does the winner advance the state of the art in AI) or for making a technical mistake (e.g., saying an NP-complete problem can't be solved in a finite amount of time). Points will be taken off for incomplete or faulty reasoning. Missing the point or not really answering the question will result in at least half the points off.

Problem 1.1 [10 points] *The Turing Test*
Do the problem below, but only answer the third question: "Can you think of new objections ... ?"

1.2 Read Turing's original paper on AI (Turing, 1950). In the paper, he discusses several objections to his proposed enterprise and his test for intelligence. Which objections still carry weight? Are his refutations valid? Can you think of new objections arising from developments since he wrote the paper? In the paper, he predicts that, by the year 2000, a computer will have a 30% chance of passing a five-minute Turing Test with an unskilled interrogator. What chance do you think a computer would have today? In another 50 years?

Problem 1.2 [10 points] *Are reflexes intelligent?*
Answer the following questions:

Are reflex actions (such as flinching from a hot stove) rational? Are they intelligent?

Problem 1.3 [10 points] *Surely computers can't be intelligent ...*
Answer the following question:

1.11 "Surely computers cannot be intelligent—they can do only what their programmers tell them." Is the latter statement true, and does it imply the former?

Problem 1.4 [20 points] *State of the Art in AI*
Do the following problem. But just pick two of the eleven that interest you the most and write a paragraph for each. You might highlight one to three successes for a particular part if there are competing systems or if "solving" the problem is broad. Make sure your answers are distinct from those in R&N, Section 1.4.

1.14 Examine the AI literature to discover whether the following tasks can currently be solved by computers:

a. Playing a decent game of table tennis (Ping-Pong).
b. Driving in the center of Cairo, Egypt.
c. Driving in Victorville, California.
d. Buying a week's worth of groceries at the market.
e. Buying a week's worth of groceries on the Web.
f. Playing a decent game of bridge at a competitive level.
g. Discovering and proving new mathematical theorems.
h. Writing an intentionally funny story.
i. Giving competent legal advice in a specialized area of law.
j. Translating spoken English into spoken Swedish in real time.
k. Performing a complex surgical operation.

Problem 1.5 [20 points] *Design and build a program to pass one piece of the Turing test*
Formulate a simple cognitive model for addition of pairs of integers. Test it on a few examples. Implement it using a programming language of your choice.

Turn in: your code and some example input/output pairs.

How might you modify or use your model as part of a cognitive model of multiplication?

Notes: For this problem, you may wish to add randomness (to make mistakes at random) and timing (to simulate the amount of time humans need to add a certain number of digits). For some pointers on random numbers, see the last paragraph of the notes on Problem 1.6 below. For a timer, you can use various pause/sleep commands. For example, Matlab has a `pause(n)` *command that pauses for n seconds; Python has* `time.sleep(n)` *with the same functionality.*

You can also repeatedly query a global time function, or use the following structure to add a delay of M seconds:

```
FOR j = 0 to M
   j++;
   FOR i = 0 to BIGNUM
      i++;
```

where `BIGNUM` *is chosen large enough that it takes you computer 1 second to do this. Note that computers today are so fast you may have to use a double loop to achieve delays this long without overflowing the constant integer data type.*

Here are the results of an in-class Turing Test on this question for a previous semester (sample size: 60 total responses)

105721: 54
105,721: 1
105361: 1
105711: 1
105724: 1
115721: 2

Times given as interval (a, b] for a < time <= b
(0,5]: 1; (5,10]: 10: (10,15]: 7; (15,20]: 11; (20,25]: 10; (25,30]: 7; (30,35]: 3; (35,40]: 2; (40,45]: 3; (45, 50]: 1; (50, 55]: 2; (55-60]: 2; plus one at 120.

Problem 1.6 [30 points] *Shannon's "Mind Reading" Machine*

a. Toss a coin 32 times and record the outcome (as a string of H and T);
b. Compute the experimentally observed probability of heads over tosses 2 through 21, inclusive (20 outcomes);
c. Compute the Markov chain transition probabilities over the first 21 tosses (viz., the first 20 transitions).
d. Repeat (a)-(c) above with a sequence verbally derived from a friend not in the class who does not know the underlying model you are trying to construct.
e. Using any language you wish, implement a computer program that uses the Markov chain model to predict the final ten transitions of each data set (throws **23-32** given the values of throws **22-31**, respectively). That is, given the previous state (throw i), compute the next state (throw i+1) using the model, compare with the actual data, and tally the error function (# of wrong guesses).

Again, turn in your code and any input/output pairs used in testing. Comment on your results.

The model you compute in part (c) should be a FIXED model computed ONCE for the ENTIRE test data set (all first twenty transitions).

The states of your Markov chain will be H and T, representing that the current toss is heads or tails, respectively. The four transition probabilities are P(H|H), P(T|H), coming out of H and going to H and T, respectively; plus P(H|T), P(T|T), coming out of T and going to H and T, respectively. These are computed as follows:

$$P(H|H) = \#(HH) / [\ \#(HH) + \#(HT)\ ]$$
$$P(T|H) = \#(HT) / [\ \#(HH) + \#(HT)\ ]$$
$$P(H|T) = \#(TH) / [\ \#(TH) + \#(TT)\ ]$$
$$P(T|T) = \#(TT) / [\ \#(TH) + \#(TT)\ ]$$

Thus, P(H|H)+P(T|H)=1 and P(H|T)+P(T|T)=1.

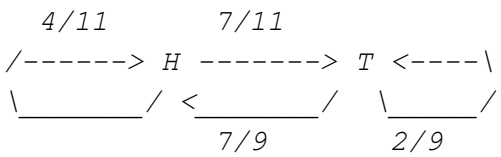
For example, consider the following data set of 31 tosses:

H H H T T H T H T H H T H T H H T H T T H | H H T H T H T H T H

The first transition is HH, the second HH, the third HT, ..., the twentieth TH.

Counting up, the number of HH tranisitions in the first twenty is #(HH) = 4.
Counting up, the number of HT tranisitions in the first twenty is #(HT) = 7.
Counting up, the number of TH tranisitions in the first twenty is #(TH) = 7.
Counting up, the number of TT tranisitions in the first twenty is #(TT) = 2.

This would lead to the model:



These may be computed by hand. ***You do not need to write a program to compute these!***

Hint: for the above programs, you may need a function that picks/returns a random number between 0 and 1. Matlab has `rand` *and Python has* `random.random()`*. Other programming languages might have commands/classes that produce a random integer,* `R`*, from 1 to some integer* `LARGE`*. You can convert this by setting* `r=R/LARGE`*.*

Specifically, for the model above, if the last toss was H, you pick a random number in the interval [0, 1]. If it is less than 7/11, you guess T; else, you guess H.
