CS601: **Homework 1** Due in class Thursday, Jan. 31, 2019

Cooperation: Students should talk to each other about the subject matter of this class and help each other. It is fine to discuss the problems and ask questions about them. I encourage such questions in class and office hours. However, there is a line past which you must not go, e.g., sharing or copying a solution is not okay and could result in failure. If a significant part of one of your solutions is due to someone else, or something you've read then **you must acknowledge your source!** Failure to do so is a serious academic violation, likely to result in failure of the course or worse. Furthermore, **all solutions must be written by yourself, in your own words**. You may get an idea from somewhere or someone and acknowledge that, but you must still understand it and explain it yourself. A copied solution, even with the source acknowledged will be considered plagiarism. The exception is if it is in quotation marks and cited specifically. But in this case, don't bother because you won't get credit for quoting someone else's solution.

Furthermore, **please do not look up answers to problems, especially not on the web**. The only appropriate places to look things up are in the text and your notes, and the notes I post. To do a problem, ask for clarifications until you understand the problem. Try it on small examples, play with it, make sure you understand the problem or ask a simple question about a small instance of the problem.

General Strategy for doing the homeworks: My desire for the homeworks is that they help you **understand** the concepts from lecture and readings, i.e., that these concepts not only seem believable, but you can employ them.

For almost all the problems I give, you should be able to look at a few tiny examples, and try to solve the problem on those examples. If you can do it for the tiny examples, then you are part way to giving a rule that will solve the problem in general. If you cannot solve or are confused by what the problem means on one of your small examples, then that would be a great time to ask a question about it. Please be careful not to ask a question on Piazza that might give away a partial solution to a hw problem. When in doubt, ask the question privately and if it's fine, I will open it up so everyone can see the question and its answer.

Problems:

1. We will think of a Turing Machine input as a binary string, $w \in \{0,1\}^*$, which appears between the left marker, "$\triangleright$" and the infinite string of blanks, $\sqcup$. The following problems show that we can ignore the type of the input: if we want to think of the input as a binary string, w, that's what it is. If we want to think of it as a natural number, fine, it's $\zeta(w)$. If we want to think of it as a pair of natural numbers, fine it's $(L(\zeta(w)), R(\zeta(w)))$, etc.

   (a) Show that the following pairing function given by Cantor is a 1:1 correspondence between $\mathbf{N} \times \mathbf{N}$ and $\mathbf{N}$:
   $$P(i,j) \quad \mapsto \quad \frac{(i+j)(i+j+1)}{2} + i$$
   [Hint: look at enough examples to get an understanding of what this function does. Then give a clear, simple proof that this map is indeed 1:1 and onto. I don't care how formal you are as long as your argument is very clear and convincing. It may help to recall that $\sum_{i=1}^{n} i = n(n+1)/2$. You might want to show the existence of the "inverses" of the pairing function. These are functions $L, R : \mathbf{N} \to \mathbf{N}$ with the properties that
   $$\forall i,j,k \in \mathbf{N}\, (P(L(k), R(k)) = k \quad \& \quad L(P(i,j)) = i \quad \& \quad R(P(i,j)) = j)$$

(b) Construct a simple, easy to compute 1:1 and onto map, ζ, from the set of binary strings to the naturals, i.e., $\zeta : \{0,1\}^\star \underset{\text{onto}}{\overset{1:1}{\to}} \mathbf{N}$. [If you are not sure what "**N**" is, check the 601 symbol table: `https://people.cs.umass.edu/~immerman/cs601/symbolTable.pdf`.]

In Lecture 2 we defined partial recursive functions, total recursive functions, recursive sets, r.e. sets and co-r.e. sets. We showed that all Turing machines can be listed out as $M_0, M_1, \ldots, M_n, \ldots$, where the subscript, n, is a natural number, which may also be thought of as a binary string, $w = \zeta^{-1}(n)$, which is a binary encoding of the the program that M_n runs, i.e., n is the program.

As Turing proved, there is a universal TM, U, with the property that for all $i, j \in \mathbf{N}$, $U(i,j) = M_i(j)$, i.e., the universal machine on input (i,j) exactly simulates M_i on input j.[1]

We also defined $W_i = \mathcal{L}(M_i) = \{w \mid M_i(w) = 1\}$, i.e., the set accepted by TM M_i is the ith r.e. set. The set of all r.e. sets is thus r.e. $= \{W_0, W_1, \ldots\}$.

We defined the diagonal set, $K = \{i \mid i \in W_i\}$. We proved that K is r.e., but that it's complement, $\overline{K} = \{i \mid i \notin W_i\}$ is not r.e.

Please read the notes for lectures 1 and 2 posted on the syllabus page.

Even though the proofs so far are reasonably simple, the concepts are deep and take a while to absorb. The goal of this homework is to help you do that.

2. Let $A_{\text{TM}} = \{(i,j) \mid M_i(j) = 1\}$.

   (a) Show directly from the definition of r.e., that A_{TM} is r.e., i.e., that $p_{A_{\text{TM}}}$ is a partial recursive function.

   (b) Show from the fact that $\overline{K}$ is not r.e., that A_{TM} is not recursive. Do this by contradiction: assume that A_{TM} is recursive, i.e., you have a TM, M_A, that computes $\chi_{A_{\text{TM}}}$. Next modify M_A to build a TM that computes χ_K.

3. (Where the name "recursively enumerable" comes from.)

   (a) Show that every finite set, F, is r.e.

   (b) Suppose that W_i is infinite. Given M_i, show how to build a TM computing a total recursive function, f_i, such that f_i enumerates W_i, i.e.,

$$W_i = \{f_i(0), f_i(1), f_i(2), \ldots\}$$

   Can you always construct f_i so that it is a 1:1 function?

[1]Note that the text uses slighly different notation in that they give U an extra first argument consisting of the number of steps that the simulation of $M_i(j)$ should run for.