1. We consider two different languages that in some sense represent strings representing correct additive equalities. You can assume that the input is on a one-way read-only tape. Show that one of these problems can be solved by a finite state machine. Show from first principles (so no using results like the Pumping Lemma) that one of these problems can not be solved by a finite state machine. To show impossibility, you should assume that there is a finite automata $M$ with $k$ states that accepts this language. Then consider arbitrarily large strings of some particular type, and argue $M$ has to mess up on some string of length greater than $k$.

(a) The language consists of strings over the symbols: 0, 1, + and =. The languages consists of strings of the form $x + y = z$, where natural numbers $x$, $y$ and $z$ encoded in binary in the standard way, and where $x$ plus $y$ is indeed equal to $z$. So for example the string $10100 + 111 = 11011$ of 11 symbols is in the language.

(b) The language is a string of the following 8 symbols (so each line is a different symbol)

\[
\begin{align*}
[000] \\
[001] \\
[010] \\
[011] \\
[100] \\
[101] \\
[110] \\
[111]
\end{align*}
\]

where the first column plus the second column is equal to the third column. So to reuse our example from the previous subproblem, the 5 symbol input

\[
[101][001][110][011][011]
\]

is in the language because $10100 + 111 = 11011$.

So this shows that whether one whether a finite state machine can add, depends on the definition of notion of addition that you use.

Due Friday January 12 (You need not use LaTeX, your solutions may be handwritten.)

2. Problem 1.1 from the text. That is, construct a complete formal description of a Turing machine (using the formalization given in section 1.2) that adds two binary numbers (you need not do multiplication). Assume that input is of the form NUMBER1#NUMBER2, e.g. 1011#110. You should write your output on a second tape.

Due Friday January 12 (You need not use LaTeX, your solutions may be handwritten.)

3. Read the description of the Post Correspondence Problem

https://en.wikipedia.org/wiki/Post_correspondence_problem
including the sketch of the proof that there is no algorithm to solve this problem. The article ends with “There are a number of details to work out, such as dealing with boundaries between states, making sure that our initial tile goes first in the match, and so on, ..”

Explain how to deal with these two details, that is, how to deal with boundaries between states, and how to make sure that the initial tile goes first. List any other details that you think need to be dealt with, and explain how they can be dealt with.

Due Wednesday January 17.

4. (a) (warm up) Show by reduction from the Halting Problem that there is no Turing machine that takes as input a Turing machine $M$, and determines whether the language $L(M)$ accepted by $M$ is the empty language.

(b) (warm up) Show by reduction from the Halting Problem that there is no Turing machine that takes as input a Turing machine $M$, and determines whether the language $L(M)$ accepted by $M$ is the language of every string over the input alphabet.

(c) (warm up) Show by reduction from the Halting Problem that there is no Turing machine that takes as input a Turing machine $M$, and determines whether the language $L(M)$ accepted by $M$ includes the string 11110.

(d) Let $P$ be some property of languages. Further assume there is a Turing machine $M_1$ that accepts a language $L_1$ that has property $P$, and a Turing machine $M_2$ that accepts a language $L_2$ that does not have has property $P$. Show by reduction from the Halting Problem that there is no Turing machine that takes as input a Turing machine $M$, and determines whether the language $L(M)$ accepted by $M$ satisfies property $P$.

(e) Explain why the first three subproblems are consequences of the fourth subproblem.

Due Friday January 19. Note that some students may find the fourth subproblem somewhat tricky.