CSEP531 Homework 1 Solution

Problem 1

Idea: We knew (in class) how to add 1 into a binary number. We can utilize this fact to build a counter. Starting with 0, we add 1 to the counter whenever we "do something". In this question, that "something" is to move the head of the output tape to the right. We stop when the value of the counter equals the value of the input.

Description: The TM we will build has 3 tapes: the INPUT, OUTPUT and WORK tapes. It works as follow. First, it writes a 0 on the WORK tape. Then, it repeats the following until the number on the WORK tape matches the number on the INPUT tape (this can be easily checked at each step): move the head of the OUTPUT tape to the right and add 1 to the number on the WORK tape.

Problem 2

Idea: We can enumerate all the quadratic equations with integer coefficients in the same way as enumerating all the rational numbers. Now, to enumerate all the roots of these equations, we only need to modify the enumeration process so that each time, instead of enumerating the equations, we enumerate the proof.

Proof: We can build the bijection from N to the set of all roots in question as follow

```
let i := 1
for all integers k = 1,2,...:
  for each integers a,b,c such that |a| + |b| + |c| = k:
    let x and y be the roots of ax^2 + bx + c = 0
    if x has not appeared before, set f(i) to x and set i := i + 1
    if y has not appeared before, set f(i) to y and set i := i + 1
```

The proof that f is a bijection is intuitive, and is left as an exercise.

Problem 3

Idea: A diagonalization similar to that in the proof that $[0, 1]$ is uncountable will work. Else, we can also show that the set of such functions is at least as large as the set of real numbers between 0 and 1.

Proof: Assume that the set of functions from $\{0, 1\}^*$ to $\{0, 1\}$ is countable, i.e. we can enumerate them as $f_1, f_2, \ldots$. Consider the following function f. On a sequence starting with a 0, f returns arbitrarily. On a sequence s starting with a 1, let n be the number whose binary representation is s. Then $f(s) = 1 - f_n(s)$. Since the set of functions from $\{0, 1\}^*$ to $\{0, 1\}$ is countable, there is some m such that $f = f_m$. Let s' be the binary representation of m. Then we have: $f_m(s) = 1 - f_m(s)$, which is a contradiction.

Problem 4

Idea: We want to design a machine M that decide L, given the machines M' and M'' that decide L' and L'' respectively. Given an input w, if we know where to break it into two pieces w' and w'', then we can

simulate M' and M'' on w' and w'' respectively. The problem is we don't know where to break w; but we can certainly try all of them, since there are at most $|w| + 1$ ways to break w into w' and w''.

Proof: The pseudo code of M is as follow

```
for all the way to break w into w' and w'':
  simulates M' on w', if it accepts:
    simulates M'' on w'', if it accepts:
      accept w
reject w
```

Extra credit

Idea: We need to show that for each program P in the given language, there is a Turing machine M that compute the same function in similar amount of time. Note that we don't need a single machine that emulate all programs, but one machine for each particular program. For each program, the number of labels is finite. Thus, a TM can be built by mapping:

- labels to states
- the array A to the TM's tape
- $\square$ to blank, 0 and 1 (the language's symbols) to 0 and 1 (the TM's symbols).
- "If A[i] equals σ then *cmds*" to "If the current symbol is $map(\sigma)$ then $map(cmds)$" where $map(cmds)$ is defined as follow:
  1. "Set A[i] to τ" is mapped to "Write $map(\tau)$".
  2. "Goto *label*" is mapped to "Go to state $map(label)$".
  3. "Increment i by 1" is mapped to "Move the head to the next cell on the right".
  4. "Decrement i by 1" is mapped to "Move the head to the next cell on the left".
  5. "Output b and halt" is mapped to "Write $map(b)$ and halt".

Proof: It is intuitive that the TM M constructed as above runs parallely with the program. Thus, it computes the same function in the same amount of time. However, to prove this rigorously is a tedious work. Let the configuration of P after t steps $C_t(P)$ be the content of A, the value of i and the label P is currently at and note that $C_t(P)$ completely determines $C_{t+1}(P)$. Let the configuration of M after t steps $C_t(M)$ be the content of the tape, the position of the head and the state M is in. Again, $C_t(M)$ completely determines $C_{t+1}(M)$. Now, we will need to prove the following statement

"At any time t, $C_t(M)$ is $map(C_t(P))$".

This can be proved by induction on t. At the beginning ($t = 0$), the statement holds by construction. Assume that at step t, $C_t(M)$ is $map(C_t(P))$; we prove that $C_{t+1}(M)$ is $map(C_{t+1}(P))$. The proof is merely a case by case analysis on which action P and M will take at this time; which is left as an exercise.