Lecture #4: Modelling Computation of Functions III
Exercises and Review

Questions for Review

1. What is an $\mathcal{S}$-program?
   (a) Describe the ***variables*** that can appear in an $\mathcal{S}$-program, as precisely as you can.
   (b) Describe the ***labels*** that can be used in an $\mathcal{S}$-program, as precisely as you can.
   (c) Describe the ***statements*** that can be used in an $\mathcal{S}$-program, as precisely as you can.
2. What is a ***macro***? What is a ***pseudo-instruction***? There might be any formal definitions of these in the lecture notes, so you should answer these questions in your own words, based on lecture material (including examples of each of these).
3. Describe ***macro expansion:***
   (a) What is the goal of this process?
   (b) Why is this process useful? (What does it help us to prove?)
   (c) Describe the details of this process, as precisely as you can.
4. How (or why) is the programming language $\mathcal{S}$ useful? What is it ***not*** useful for?
5. Define (or describe) each of the following, as precisely as you can.
   (a) an ***$\mathcal{S}$-program***
   (b) the ***length*** of an $\mathcal{S}$-program
   (c) the way that ***numbers*** will be assigned to statements in an $\mathcal{S}$-program
   (d) a ***state***
   (e) a ***snapshot*** (or ***instantaneous description***)
   (f) a ***computation*** of an $\mathcal{S}$-program
   (g) the *set* of ***functions*** computed by an $\mathcal{S}$-program
   (h) what it means for a (partial or total) function $f : \mathbb{N}^k \to \mathbb{N}$ to be ***$\mathcal{S}$-computable***

   How, or why, will these formal definitions be useful, later on in this course?

6. How are the set of ***primitive recursive functions*** and the set of ***$\mathcal{S}$-computable total functions*** related? How is (part of) this relationship proved?
7. How are the set of ***μ-recursive*** partial or total functions and the set of ***$\mathcal{S}$-computable*** partial or total functions related? How is (part of) this relationship proved?

Practice Problems

These problems will not be discussed during the lecture, and solutions for these problems will not generally be made available. They can be used as "practice" problems that can help you to practice skill considered in the lecture presentation for Lecture #4.

1. Consider the macro (and corresponding program) for an ***assignment statement*** included in the online notes.

   (a) Write down the program $\mathcal{P}_{Y \leftarrow X_1}$ corresponding to the pseudo-instruction

   $$Y \leftarrow X_1$$

   (b) Note that — if you only performed the ***macro expansion*** process *once* — the resulting program is not a program in $\mathcal{S}$, because it still includes pseudo-instructions. Carry out the macro expansion process again, as many times as required, to obtain a program in $\mathcal{S}$ (that does not include any pseudo-instructions, at all) that *also* corresponds to the macro $Y \leftarrow X_1$.

2. Consider the following $\mathcal{S}$-program.

1.		IF $X_1 \neq 0$ GOTO A_1
2.		$Z_1 \leftarrow Z_1 + 1$
3.		IF $Z_1 \neq 0$ GOTO A_2
4,	$[A_1]$	$X_1 \leftarrow X_1 - 1$
5.		$Y \leftarrow Y + 1$
6.		IF $X_1 \neq 0$ GOTO A_1

   (a) Show the ***computation*** (that is, sequence of ***instantaneous descriptions*** or ***snapshots***) corresponding to an execution of this program with input $X_1 = 2$.

   (b) Describe the functions $f_1 : \mathbb{N} \rightarrow \mathbb{N}$ and $f_2 : \mathbb{N}^2 \rightarrow \mathbb{N}$ computed by this $\mathcal{L}$-program.

3. Suppose that W, X, and Z are distinct variables.

   (a) Write a program $\mathcal{P}_{W \leftarrow X^Z}$ that corresponds to the pseudo-instruction for exponentiation,

   $$W \leftarrow X^Z.$$

   In other words, if your program is executed when the initial values of X and Z are n and m, respectively, where $n, m \in \mathbb{N}$, then your program should eventually halt. On termination the variables W, X and Z should have values n^m, n and m respectively — where you may assume that $n^0 = 1$ for all $n \in \mathbb{N}$ and that $0^m = 0$ for very *positive* integer m.

   Hint: Start by studying the programs in the lecture notes and the lecture presentation, and make use of the pseudo-instructions and macros that have been introduced. You should find that it is reasonably easy to modify (at least one) of the programs in the notes in order to solve this problem.

   (b) Sketch a proof of the correctness of your program. Once again, you should be able to do this by studying one (or more) of the examples that are lecture notes and modifying them so that they accurately describe the behaviour of your program.