FLAC Assignment 8

Exercise 1. Show that the question of whether a Turing Machine halts on a blank tape is undecidable.

Exercise 2. A *queue automaton* is like a pushdown automaton except that the stack is replacd by a queue. A *queue* is a tape allowing symbols to be written only on the left-hand end and read only at the right-hand end. Each write operation (we'll call it a *push*) adds a symbol to the left-hand end of the queue and each read operation (we'll call it a *pull*) reads and removes a symbol at the hand-hand end. As with a PDA, the input is placed on a seperate read-only input tape, and the head on the input tape can move only from left to right. The input tape contains a cell with a blank symbol following the input, so that the end of the input can be detected. A queue automaton accepts its input by entering a special accept state at any time.

Show that the question of whether a deterministic queue automaton (DQA) halts on a tape is undecidable. (Hint: Show that a language can be recognized by a DQA iff the language is Turing-recognizable.)

Exercise 3.

a. We have defined *recursively enumerable (r.e.)* languages as those that can be recognized by a Turing Machine. Using this definition, show that every recursively enumerable language B has a computable function f that enumerates B, i.e., that

$$B = \{f(n) \mid n \in \mathbb{N}\} = \{f(0), f(1), \ldots\}$$

Note that a similar result in proved in your textbook. Your proof may closely follow the book's proof, but do not simply copy the proof from the book. Rather, after you absorb the main idea from the book, close the book and write out the proof by yourself.

b. Show that if a language B is recursively enumerated by a *strictly increasing* computable function f (so $n < m \implies f(n) < f(m)$), then B is recursive (decidable).

c. Use these facts to prove that every infinite r.e. language has an infinite recursive subset.

Exercise 4.

a. Give a Turing Machine that computes the function $f(n) = 2^n$ in unary notation. (Give a low-level implementation description and indicate the number of states required. You need not explicitly draw a state diagram or give a formal transition table.) You may refer to the Double function from the previous homework in your description. (I.e., you don't need to describe how to implement Double again.)

b. Referring to your solution in Part (a) above, give a low-level implementation description of a Turing Machine that computes $f(n) = \underbrace{2^{2^{\cdot^{\cdot^{\cdot^{2^n}}}}}}_{n \text{ copies of } 2}$. Indicate the number of states.

c. What does your answer in the above part tell you about the Busy Beaver function?