Cpt S 516 Homework # 1

Convention: In problems 1,2,3, Turing machines are nondeterministic Turing machines. In all the other problems, Turing machines are deterministic Turing machines.

1. (standard, 20pt) A "change" is an event that some cell's content is overwritten by a different symbol. Notice that, if initially a cell is empty, a "change" occurs when it holds a non-null symbol. Turing machine M is *10-change* if during any computation of M on any input word, there are at most 10 changes.

(1). What is the smallest class of languages that 10-change Turing machines can accept? Prove your answer.

(2). (hard) A Turing machine M is *10-change-per-cell* if during any computation of M on any input word, there are at most 10 changes occurring at each cell. Prove that any deterministic Turing machine can be simulated by a 10-change-per-cell one.

2. (standard, 10pt) A "turn" of the head of a Turing machine means that the head changes its direction from moving to the right to moving to the left, or vice versa. For instance, the following sequence of moves of the head:

$$R, R, R, R, S, S, S, S, L, L, L, \cdots$$

constitutes one turn (where L="move to the left", R="move to the right", and S="stay"). A Turing machine M is *1-turn* if during any computation of M on any input word, the head of M makes at most one turn. What is the smallest class of languages that 1-turn Turing machines can accept? Prove your answer.

3. (standard, 10pt) In the standard model of TMs, the tape head moves can be: L (left), R (right), S (stay). Now, I modify L to J, where J means that the tape head jumps to the left end of the tape. So a TM under the new model can no longer move one cell to the left. If you really want to move to the left, the only choice is to jump to the start of the tape. Call this kind of TMs as *jump-TMs*. Show that any TM can be simulated by a jump-TM.

4. (standard, 10pt) how to simulate two counters using one (FIFO) queue? Give me the procedure.

5. (A research problem, but I know the answer. 15pt) As we talked in class, a program with two integer variables are universal. Now, we consider a spcial form of two variable programs. Let $G = (V, E)$ be a directed graph, where V is a finite set of *nodes*, and $E \subseteq V \times V$ be the set of *(directed) edges* (arcs). In particular, we identify a node as *the initial node*, and a node as *the final node*. Let x and y be two non-negative integer variables. Further, we decorate each edge with one of the following *instructions*:

$x := x + 1$;
$x := x - 1$;
$x == 0?$;
$y := y + 1$;
$y := y - 1$;
$y == 0?$;

The result is called a *decorated graph* (we still use G to denote it). The semantics of a decorated graph is straightforward. It executes from the initial node with x and y being 0, then walks along the graph. G can *walk an edge* (v, v') if all of the following conditions are satisfied:

- if the edge is decorated with instruction $x := x + 1$, the new value of x is one more than the old value, and y is unchanged.
- if the edge is decorated with instruction $x := x - 1$, the new value of x must be non-negative and is one less than the old value, and y is unchanged.
- if the edge is decorated with instruction $x == 0?$, the value of x must be 0.
- the case for instructions $y := y + 1, y := y - 1, y == 0?$ are similar.

If at a node, G has more than one edge that can be walked, then G nondeterministically chooses one. If at a node G has no edge that can be walked, then G crashes (i.e., do not walk any further). We further require that each variable behaves as follows: a number of increments followed by a number of decrements – but after that, it will NOT not be incremented again. We say that a decorated graph G is *terminating* if G can walk from an initial node to a final node. Show me an algorithm that answers (yes/no) whether G is terminating or not.

6. (very hard, 15pt) Let $G = (V, E)$ be a directed graph, where V is a finite set of *nodes*, and $E \subseteq V \times V$ be the set of *(directed) edges* (arcs). In particular, we identify a node as *the initial node*, and a node as *the final node*. Let x and B be two non-negative integer variables. Further, we decorate each edge with one of the following four *instructions*:

$x := x + 1$;
$x := x - 1$;
$x == 0?$;
$x == B?$;

The result is called a *decorated graph* (we still use G to denote it). The semantics of a decorated graph is straightforward. It executes from the initial node with some non-negative values of B and x (satisfying $0 \leq x \leq B$), then walks along the graph. G can *walk an edge* (v, v') if all of the following conditions are satisfied:

- if the edge is decorated with instruction $x := x + 1$, the new value of x must satisfy $0 \leq x \leq B$.
- if the edge is decorated with instruction $x := x - 1$, the new value of x must satisfy $0 \leq x \leq B$.
- if the edge is decorated with instruction $x == 0?$, the value of x must be 0.
- if the edge is decorated with instruction $x == B?$, the value of x must equal the value of B.

If at a node, G has more than one edge that can be walked, then G nondeterministically chooses one. If at a node G has no edge that can be walked, then G crashes (i.e., do not walk any further). It is noticed that B can not be changed during any walk. However, its initial value can be any non-negative integer. We say that a decorated graph G is *terminating* if there are non-negative initial values for B and x (satisfying $0 \leq x \leq B$) such that G can walk from an initial node to a final node. Show me an algorithm that answers (yes/no) whether G is terminating or not.

7. (A research problem, 20pt) Computing has been evolving from a standalone device to a massive network of devices. This results in a distributed system running over, sometimes, millions of individual computing devices;

e.g., sensor network in battle field, P2P, etc. How would you define, mathmatically, the computing power of such systems? (Traditional Turing theory adopts a hierachy of computing power such as: finite automata, pda, TM; in parallel, we have regular, context-free, recursive, r.e., for the problems hierachy.)