1. Let $M$ be a random access machine, whose time complexity according to the logarithmic-cost model is $T(n)$. At the lecture we sketched a proof showing that $M$ can be simulated by a Turing machine $M'$ whose time complexity is $O(T(n)^3)$.

Sketch how TM $M'$ would simulate the RAM instruction store *2 in time $O(T(n))$. Assume that at the start the contents of AC is stored, in binary, on the auxiliary tape of $M'$.

2. Give a yes/no answer, with a short justification, to the following questions:

(a) If some problem of class $\mathcal{NP}$ is solvable in polynomial time, are then (i) all problems of class $\mathcal{NP}$, or (ii) all NP-complete problems solvable in polynomial time?  

(b) If some NP-complete problem is solvable in polynomial time, are then (i) all problems of class $\mathcal{NP}$, or (ii) all NP-complete problems solvable in polynomial time?  

(c) If some problem of class $\mathcal{NP}$ requires super-polynomial time, do then (i) all problems of class $\mathcal{NP}$, or (ii) all NP-complete problems require super-polynomial time?  

(d) If some NP-complete problem requires super-polynomial time, do then (i) all problems of class $\mathcal{NP}$, or (ii) all NP-complete problems require super-polynomial time?

3. Present an algorithm that returns the truth value of a Boolean formula, when the formula is given as an expression tree and the truth assignment in a suitable form, say, in an array. Analyze the time complexity of the algorithm. (If you do not know what an "expression tree" is, try googling the Net for it on.)

4. We say that a Boolean formula is refutable, if some truth assignment makes it false. Let’s define the language

$$\text{REFUTABLE} = \{ w \mid w \text{ is a refutable Boolean formula}\}.$$  

Show that REFUTABLE is NP-complete. (Hint: the proof is easy.)

5. A Boolean formula is in disjunctive normal form (DNF), if it is of the form

$$C_1 \lor \cdots \lor C_m \quad (m \geq 1),$$
where each $C_i$ is a conjunction $(l_{i_1} \land \cdots \land l_{i_k})$ of literals. Sketch a deterministic algorithm that tests whether a DNF formula is satisfiable, in time that is polynomial with respect to the length of the formula. (Hint: This is not difficult either; When does a DNF formula become true? When is a conjunction of literals satisfiable?)

6. An instance of the Partition problem consists of a set of items $A = \{a_1, \ldots, a_m\}$, each with a weight $w(a_i) \in \mathbb{N}$. The task is to test whether the set can be divided in two equally heavy subsets $A' \subseteq A$ and $A - A'$, that is,

$$\sum_{a \in A'} w(a) = \sum_{a \in A - A'} w(a).$$

Partition is known to be NP-complete. Based on this, give an alternative proof for the NP-completeness of the (decision version of the) Knapsack problem discussed at the lecture.