Computational Models, Spring 2014 Exrecise #6

Turing Machines, $\mathcal{P}$ and $\mathcal{NP}$

1. Show that the following languages are NP-Complete
   (a) $Partition := \{x_1, \ldots x_n \mid \exists S \subseteq [1 \ldots n], \sum_{i \in S} x_i = \sum_{i \notin S} x_i\}$
   (b) $XS := \{x_1, \ldots x_n \mid \exists S \subseteq [1 \ldots n], \sum_{i \in S} x_i + |\bar{S}| = \sum_{i \in \bar{S}} x_i + |S|\}$
2. Show that both of these decision problems are in $\mathcal{P}$
   (a) DNF-Satisfability: Given a DNF formula is it satisfiable.
   (b) CNF-Tautology: Given a CNF formula, is it a tautology (a tautology is a formula which is true in every possible assignment)?
3. Consider the following algorithm to solve the vertex cover problem. First, we generate all size-k subsets of the vertices. There are $O(n^k)$ of them. Then we check whether any of the resulting subgraphs is complete. Why is this not a polynomial-time algorithm?
4. For the following decision problems, determine whether they are in $\mathcal{P}$ or in $\mathcal{NPC}$ (assuming $\mathcal{P} \neq \mathcal{NP}$). Prove your answer.
   (a) Input: sets $A_1 \ldots A_n$, and a number k.
   Question: does there exist a set C of size k, such that for every $1 \leq i \leq n$ $A_i \cap C \neq \emptyset$?
   (b) Input: a 3CNF formula ψ Question: does there exist an assignment that satisfies ψ and gives **True** for exactly 10 variables?
   (c) Input: a 3CNF formula ψ
   Question: do there exist at least two assignments that satisfy ψ?
   (d) Input: graph G.
   Question: does there exist a Hamiltonian path in G (between any pair of vertices)?
   (e) Input: graph G and a number k.
   Question: does there exist a simple path in G of length $\geq k$?
   (f) Input: graph G and a number k.
   Question: is there a Vertex-Cover S in G of size k and an IndependentSet, T, of size $\frac{k}{2}$, such that $T \subseteq S$?
5. Prove that if $\mathcal{P} = \mathcal{NP}$ then every language in $\mathcal{P}$, except $\emptyset$ and Σ^* is $\mathcal{NPC}$. Why can't $\emptyset$ and Σ^* be $\mathcal{NPC}$.
6. Consider the following language:

   $$L = \{\#1^n\#x_1, \ldots x_n \mid \exists i, j \ s.t. x_i = x_j\}$$

   . A possible way to implement a Turing Machine that accepts the language is as follows: "Choose (non-deterministically) two indices i and j $(0 < i, j \leq n)$ and write them on a second tape. Now check (using a deterministic TM) if (1) $i \neq j$ and (2) $x_i = x_j$."

   In this exercise we will *formally* implement the first part: On a *two-taped, non-deterministic* Turing Machine with input $\#1^n\#$ $(n > 1)$, choose non-deterministically i and j such that $0 < i, j \leq n$ and write on the second tape $1^i\#1^j$ and accept.

   If this helps you, you may use a model that allows the head to stay put as well as moving right and left.
7. We wish to understand the *encoding* of graphs

(a) Suggest an efficient way to encode an undirected graph $G = (V, E)$ as a string $< G >$. Describe the complexity of your encoding as a function of edges and vertices. Draw several graphs and demonstrate the encoding on the graphs.

(b) Write a program (python or scheme) that takes as an input an encoding $< G >$ of a graph and an integer k and returns true if the sum of the degrees of the vertices equals to k.