Analysis of Algorithms: Assignment 10
Due date: April 22 (Thursday)

Problem 1 (3 points)
Assume that all characters in a pattern $P[1..m]$ are *distinct,* and you need to find all occurrences of P in a text $T[1..n]$. Write an "accelerated" version of NAIVE-STRING-MATCHER, which solves this problem in $O(n)$ time.

Problem 2 (3 points)
Write an algorithm that looks for a given $m \times m$ pattern in an $n \times n$ array of characters, based on the Rabin-Karp method. The pattern may be shifted vertically or horizontally within the $n \times n$ array, but it cannot be rotated.

Problem 3 (4 points)
Using Figure 34.6(a) in the book as a model, draw a string-matching automaton for the pattern `aabac`; that is, your automaton must accept the strings with the suffix `aabac`.

Problem 4 (bonus)
The optional problem is **very hard**, *although its solution is surprisingly straightforward. If you solve this problem, then you get* **6 bonus points** *toward your final grade for the course. As usual, you cannot submit it after the deadline.*

Imagine that some researcher has just proved that P=NP, and we have all reasons to trust her, even though we have not yet seen the proof. Then, we know that every NP problem has a polynomial-time solution; in particular, there exists a polynomial algorithm that solves the Hamiltonian-Cycle problem.

Surprisingly, this knowledge enables us to implement an actual program that finds hamiltonian cycles in polynomial time. Your task is to describe an algorithm that underlies such a program. If the P=NP result is correct, then your algorithm must always run in polynomial time. On the other hand, if P≠NP, then the algorithm may take exponential time.

Hints:

- You do not need to utilize any special properties of the Hamiltonian-Cycle problem. The idea underlying the solution is applicable to any NP problem.
- Even if P=NP and your algorithm runs in polynomial time, you cannot use it in real-life applications, because a huge constant factor makes it impractically slow.