0. Basic Concepts.
This problem is only for self-study only; do not hand in. This problem checks your knowledge of some basic facts used in this course. Note: these do not cover all the needed background; but if you do not know how to answer some of these, it is time to review the Appendix of the textbook (which will be useful no matter what).

1. \( \ln(e^x) = ? \) \( \log_{10}(1000) = ? \) \( \log_2(1024) = ? \)
2. \( \sum_{i=1}^{n} i = ? \)
3. \( \sum_{i=1}^{n} \frac{1}{i} = ? \)
4. What is approximately \( \sum_{i=1}^{n} \frac{1}{i^2} \)?
5. \( \sum_{i=1}^{n} \sum_{x=1}^{m} ix = ? \)
6. We define unrooted tree as a undirected, acyclic graph. How many edges are there in an unrooted tree with \( n \) nodes?

7. How many permutations are there for \{1,2,3,4,5\}? How many (un-ordered) pairs are there from \{1,2,3,4,5\}?

8. How many ways of choosing \( k \) different elements from a set of \( n \) different elements are there? Suppose we need to consider all \( k = 0 \ldots n \), and for each \( k \) we need to consider all possible subsets with \( k \) different elements from a set of \( n \) different elements. How many subsets do we need to consider?

9. Let \( x \) be a random binary variable (i.e. \( x \) can be 0 or 1), where the probability of \( x = 1 \) is 0.6. What is the expectation of \( x \) (i.e. \( E(x) \))? Let \( x \) and \( y \) be two random variables, whose expected values are 1 and 2 respectively. What is \( E(x + y) \)?

10. Let \( S \) be a set with \( n \) elements. How many distinct subsets does \( S \) have?

1 Matrix Multiplication
Given two \( n \) by \( n \) matrices \( A \) and \( B \), write a simple algorithm to compute the product of \( A \) and \( B \). Then, analyze your algorithm to find out how many additions and multiplications your algorithm will take. Here, treat each addition or multiplication of two whole integers as one operation. You do not need to come up with the most efficient algorithm; the goal of this problem is getting you started on algorithm analysis.

2 Stable Matching: A Generalization
This problem is taken from the book chapter posted on the class web page (Exercise 4 on p.23). One generalization of stable matching discussed in class is to consider the situation of National Resident Matching Program, which matches medical students to hospitals. Here, there are \( m \) hospitals and \( n \) students. Each hospitals has one or more openings for residents, and each student can only work for one hospital. We assume there are more students than the number of openings. That is, there will be students who can not find positions. Like before, each student has a ranking of (all) hospitals and each hospital has a ranking of (all) students. The problem is to find a stable assignment of students to hospitals, so that all openings are filled. We say an assignment is stable if neither of the following occurs:
1. First type of instability. There are two students, $s$, $s'$, and a hospital $h$, where $s$ is assigned with $h$, $s'$ is free, and $h$ prefers $s'$ over $s$.

2. Second type of instability. There are two students, $s$ who is assigned to hospital $h$, and $s'$ who is assigned to $h'$. But $h$ prefers $s'$ over $s$ and $s'$ prefers $h$ over $h'$.

Now show that there always exists a stable assignment by giving an efficient algorithm for this problem.

3 A tour and path problem
You and your friend Jack (a Math major) are touring an European city. This city is special: it consists of several small isles, connected by beautiful bridges. Assume you can reach an isle from any other isle by crossing bridges. Since you want to check out each of the bridges, you and Jack want to find a tour which starts from some isle and crosses each bridge at least once (even if this means you may visit some isles more than once). Moreover, to save efforts, you want to cross each bridge exactly once (you can visit an isle more than one time) if possible. After thinking about it for a moment, Jack asserts such a tour (that crosses each bridge exactly once) exists for this city because the number of bridges touching each isle is an even number.

**Part 1.** Now, as a computer science major, design an efficient algorithm for finding such a tour. That is, assuming the number of edges connecting each isle is even, your algorithm will find a tour starts from some isle and crosses each bridge exactly once. You do not need to give detailed analysis of the efficiency but you should briefly argue why the algorithm is efficient.

**Hint:** suppose we start from some isle (which one to start really does not matter), and keep going by choosing a bridge that we have not crossed before at each isle. When will this procedure stop? If by the time the procedure stops we have crossed all bridges, then we are done. But what are you going to do if not?

**Part 2: The path variation.** Now, instead finding a tour, we are only interested in finding a path that starts from some isle and crosses each bridge exactly once (but does not need to return to the starting isle). Tell me what difference it makes and how you are going to find the desired path. That is, since the problem formulation changes, the condition for existence of solution will also change; you need to tell me how the solution condition changes and how you will modify your algorithm to find such a path if exists. Also note that a tour (or cycle) can be considered as a path with the same start and ending isle, but a path usually is not a cycle.

**Part 3: The one-way bridge variation.** Suppose each bridge is a one-way bridge. In this case, tell me under which condition a tour crossing each bridge exactly once is still feasible (that is, what properties the bridges going in and out of an isle should satisfy to allow such a tour?). Moreover, suppose the tour is indeed feasible. Describe an algorithm that will find such a tour.

4 The Pancake Problem
A stack of $n$ pancakes is placed in front of you. You have a spatula which you can insert anywhere into the stack and flip over all the pancakes above the spatula. You want to arrange the pancakes in order of their diameter (they are perfectly round), and you want to use as few flips as possible. As an example suppose $n = 6$, and the pancakes are numbered 1 through 6 in order of their diameter with 1 the smallest and 6 the largest. Suppose the original order is 346215, and the left end of the sequence represents the top of the stack. In one flip I can get 643215 (by flipping the first three pancakes: 346), then in the next flip 512346, then 432156, then 123456, so four flips are enough in this case. Let $F(n)$ be the worst case number of flips needed to arrange a stack of $n$ pancakes. Find an efficient (in a worst case sense) algorithm for this problem, where efficiency is measured by the worst case number of flips. Remember that your algorithm should work for pancakes with any order. To start you off you should easily be able to show that $F(n)$ is at most $2n-2$. Next, reduce that bound a little more if you can.