Problem 1. You have one thousand $1 bills. How can you distribute them among 10 envelopes so that any amount between $1 and $1000, inclusive, can be given as a combination of these envelopes? No change is allowed, of course.

Problem 2. You are given twelve coins that are identical in appearance; either all are genuine or exactly one of them is fake. We do not know whether the fake coin is lighter or heavier than the genuine ones. You have a two-pan balance scale without weights. Each weighing would yield either the coins are =, <, or > each other in weight. You may assume < means lighter in weight and > means heavier in weight. You are required to find whether all the coins are genuine and, if not, find the fake coin and establish whether it is lighter or heavier than the genuine ones. Use decision tree (display it) model to solve the problem with the minimum number of weighings. Find the lower bound on the number of weighings.

Problem 3. Karatsuba algorithm for integer multiplication has a runtime complexity of $O(n^{1.58})$. We want to test a few multiplication algorithms along the lines of Karatsuba. You may use $\mu$ for an atomic multiplication and $\alpha$ for an atomic addition, wherever required. Please follow and answer the following questions:

1. Write a simple (without any clever tricks) recurrence equation for a multiplication algorithm that divides each $n$ digit number into 3 parts rather than 2 parts with the size of each part being $\frac{n}{3}$. We will call this TC algorithm.

2. Solve the recurrence equation for TC algorithm using the recursion tree approach as shown in class.

3. For this and the rest of the parts we will only find the Big-O runtime. What is the runtime of TC algorithm?

4. You probably noticed that TC algorithm is slower than Karatsuba’s algorithm. We will try to make it faster. While still splitting the integers into three parts, we will reduce the number of multiplications for TC algorithm by 1. What is the runtime of this algorithm with this modification?

5. What should be the number of multiplications (keep the 3 part split), so that TC algorithm becomes faster than Karatsuba algorithm? What is the runtime?

6. Now we will divide each $n$-digit number into 4 parts, rather than 3, with the size of each part being $\frac{n}{4}$. How many multiplications do we need so that the runtime of this new algorithm is faster than Karatsuba algorithm?