Lab 1

IMPORTANT: Math Problems 1 and 2 will be discussed on the second day of class—be sure to work on these before then.

Math Problem 1. Which of the following functions are increasing? eventually nondecreasing? If you remember techniques from calculus, you can make use of those.

(1) $f(x) = -x^2$

(2) $f(x) = x^2 + 2x + 1$

(3) $f(x) = x^3 + x$

Math Problem 2. Consider the following pairs and functions f, g. Decide if it is correct to say that, asymptotically, f grows no faster than g, g grows no faster than f, or both.

(1) $f(x) = 2x^2$, $g(x) = x^2 + 1$

(2) $f(x) = x^2$, $g(x) = x^3$

(3) $f(x) = 4x + 1$, $g(x) = x^2 - 1$

Problem 1. *GCD Algorithm.* Write a Java method `int gcd(int m, int n)` which accepts positive integer inputs m, n and outputs the greatest common divisor of m and n.

The rest of this lab is an exploration of the SubsetSum Problem (mentioned in the slides) and possible algorithms to solve it. Here is the statement of the problem once again: You are given a set $S = \{s_0, s_1, \ldots, s_{n-1}\}$ of positive integers and a non-negative integer k; S and k are inputs to your algorithm. Your algorithm will return "true" if there is a subset T of S such that the sum of the elements of T is precisely k, "false," otherwise.

Problem 2. *Brute Force Solution.* Formulate your own procedure for solving the SubsetSum Problem. Think of it as a Java method `subsetsum` that accepts as input S, k, and outputs a subset T of S with the property that the sum of the s_i in T is k if such a T exists, or `null` if no such T can be found. (A non-null return value can be thought of as a return of "true" and a null return value signifies "false.") Implement your idea in Java code.

Problem 3. *Greedy Strategies.* See if you can solve SubsetSum problems using the following *greedy* strategy. With a greedy strategy, at each step in an algorithm a value that is optimal *at*

that time is chosen. Decide whether the following greedy strategy works: Begin by sorting the input set S; assume that S in sorted order is as follows: $\{s_0, s_1, \ldots, s_{n-1}\}$. Initialize an empty set T; we will add elements to T as we scan S. As you scan S, if $s_0 \leq k$, put s_0 in T; otherwise leave it out. Then, if the sum of the elements of T together with s_1 is $\leq k$, then put s_1 in T; otherwise leave it out. Then, if the sum of the elements of T together with s_2 is $\leq k$, put s_2 in T; otherwise, leave it out Continue this way until every number in S has been checked.

To illustrate the algorithm, consider the following example: Suppose $S = \{3, 5, 6, 2\}$ and $k = 10$. After sorting, we have $S = \{2, 3, 5, 6\}$. We notice $2 \leq 10$, so we put 2 into T. Now $T = \{2\}$. Next we check the number 3 in S. Since $2 + 3 \leq 10$, we also place 3 into T. Now $T = \{2, 3\}$. Next we check the number 5. Since $2 + 3 + 5 = 10$, we include 5 in T as well. Now $T = \{2, 3, 5\}$. Finally we check 6. Since $2 + 3 + 5 + 6 > 10$, we reject 6. Our final value for T is $\{2, 3, 5\}$, and this is a correct solution.

For this problem, decide if this strategy always works. If not, give an example of a Subset-Sum problem for which the algorithm gives an incorrect result. If you think it does work, give an argument to support your idea.

Problem 4. You are given a solution T to a SubsetSum problem with a $S = \{s_0, s_1, \ldots, s_{n-1}\}$ and k some non-negative integer. (Recall that T is a solution if it is a subset of S the sum of whose elements is equal to k.) Suppose that s_{n-1} belongs to T. Is it necessarily true that the set $T - \{s_{n-1}\}$ is a solution to the SubsetSum problem with inputs S', k' where $S' = \{s_0, s_1, \ldots, s_{n-2}\}$ and $k' = k - s_{n-1}$? Explain. *Hint.* The sum of an empty set of integers is (by convention) equal to 0.