Assignment 1
CSC358 Winter 2012
Due Date: Feb 6th, Noon, Place TBA

Problems 1

Given two sorted arrays, A[1…n] and B[1…k], we would like to find the ith largest element in the union of the two arrays.

a) Design an algorithm to solve the problem. Your description should be accurate and concise—pseudo-code is recommended. Justify correctness.

b) Describe the running time of your algorithm using a recurrence relation, and solve it. (NOTE: To get more than half the points for (a), your algorithm should run in $O(\log(n) + \log(k))$ time—otherwise you could get at most half the points for this problem)

Problem 2

This is an implementation problem. Please implement C code for DPV 2.32 (given a set of n points on the plane, find the closest pair of points). For simplicity you can assume that your input will be < 1,000,000 points. Your algorithm, however, should be sub-quadratic (quadratic on 1 million will be very slow). You should not use any code that is not either your own, or part of the standard C libraries (though you may find the qsort C library very useful). While you will submit the code using your ECF account as "hw1", your written assignment should describe the running time of your algorithm. Your submission should include a makefile so that your program can be compiled with "make". Your program (called "closestpoints" should take x,y floating point pairs, one per line, and output the two which are closest (% is the unix prompt, # is a comment):

```
%cat >infile    #put input into infile
1.1,1000
2.2,4.5
3.3,9
5.5,11000
%closestpoints < infile   #run it
2.2,4
3.3,9
```

Note that the book proposes either an $O(n \log^2 n)$ or an $O(n \log n)$ algorithm. If you do implement an $O(n \log n)$ one, describe how you did this for up to 5 bonus points.

Problem 3

Your friend is working as a camp counselor who is in charge of organizing activities for a set of campers. One of the plans is the following mini-triathlon exercise: each contestant must swim 20 laps of a pool, then bike 10 kilometers, then run 3 kilometers. For safety reasons, the plan is to send the contestants out in a staggered fashion, via the following rule: the contestants must use the pool one at a time. In other words, first one contestant swims the 20 laps, gets out, and starts biking. As soon as this first person is out of the pool, a second contestant begins swimming the 20 laps; as soon as he or she is out and starts biking, a third contestant begins swimming, and so on. Note that participants can bike simultaneously and can run simultaneously, but at most one person can be in the pool at any time; also, contestants must complete the events in the same order—swimming, biking, running.

Each contestant has a projected swimming time (the expected time it will take him or her to complete the 20 laps), a projected biking time (the expected time it will take him or her to complete the 10kms of bicycling), and a projected running time (the expected time it will take him or her to complete the 3kms of running).

Your friend wants to decide on a schedule for the triathlon: an order in which to schedule the contestants. Let's say that the completion time of a schedule is the earliest time at which all contestants will be finished with all three legs of the triathlon, assuming they each spend exactly their projected swimming, biking, and running times on the three parts.

You are asked to give an algorithm that specifies the order in which to send people out in a way that minimizes the completion time.

Consider three different greedy strategies:

- Always send out the person with the longest total projected time first.
- Always send out the person with the shortest projected swimming time.
- Always send out the person whose sum projected time for biking and running is biggest.

For each strategy, either prove or disprove that it always produces the optimal solution.

Problem 4

Consider the problem of making change for n cents using as few coins as possible.

a) Describe a Greedy Algorithm to make change consisting of quarters, dimes, nickels and pennies. Prove that your algorithm yields an optimum solution.

b) Give a set of coin denominations for which your Greedy Algorithm does not yield an optimum solution. Your set should include a penny, so that there is a solution for every value of n.

Problem 5

When considering Huffman Codes in class, we emphasized that the code has to be prefix-free and minimal: that is, no code for one symbol can be a prefix for the code for another symbol, and no code for one symbol could be made shorter while maintaining the prefix-free property. These prefix codes were represented by full binary trees, with each symbol corresponding to a leaf. We consider two codes distinct if any of the symbols is coded by a different number of bits. For example, you cannot get a different code by simply inverting every bit.

a. How many possible distinct codes are there for an alphabet of 3 symbols? What about 4 and 5? Justify your answer.

b. Write down a recurrence that describes the total number of distinct codes for an alphabet of n symbols. Justify.