UNIVERSITY of TORONTO – CSC 373 – JUNE 22, 2006

TEST 1

Student Number: _____

Email Address: _____

Last (Family) Name(s): _____

First (Given) Name(s): _____

Tutorial (circle): M. Demko (BA3012) M. Fazel-Zarandi (BA3116)

- This test is worth 15% of your final mark.
- Answer each question directly on the test paper, in the space provided. Use the reverse side of the pages for rough work. If you need more space for one of your solutions, use the reverse side of a page and *indicate clearly the part of your work that should be marked.*
- **20% rule:** If you are unable to answer (part of) a question, you will get 20% of the marks for the (part of the) question if you write "I don't know" and *nothing else* for that part/question.

Question 1 (out of 12)	
Question 2 (out of 12)	
Question 3 (out of 16)	
Total (out of 40**)**	

Question 1 [12] Recall the Interval Scheduling Problem: given a set of requests (or intervals) $\{1, \ldots, n\}$, where each request i has starting time $s(i)$ and finishing time $f(i)$, the objective is to find a "compatible" subset S of requests with maximum cardinality.

a [2] What does it mean for a subset S of $\{1, \ldots, n\}$ to be "compatible" ?

Consider the greedy strategy that always chooses the requests *that start as late as possible*. In other words, it (i) sorts the requests into *non-increasing* order of *starting time*, and (ii) goes through the sorted list of requests, takes each request if it does not create conflict with the current (partial) solution.

b [3] Give a pseudo-code for the above algorithm.

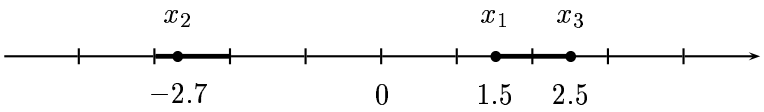
c [**7**] Does the algorithm in part **b)** always produce an optimal solution ? Give a counter-example if answer "No", and give a proof if answer "Yes".

Question 2 [12]

Consider the problem that, given a set $\{x_1, \ldots, x_n\}$ of points on the real line, determines the smallest set of unit-length closed intervals that contains all of the given points.

For example, on input $\{-2.7, 1.5, 2.5\}$, the following set of two unit intervals is an optimal solution:

$$\{[-3, -2], [1.5, 2.5]\}$$



a [5] Give the pseudo-code for a greedy algorithm that solves the above problem.

b [**2**] What is the running time of your algorithm ?

c [**5**] Prove that your algorithm always produces an optimal solution.

Question 3 [16]

The Longest Increasing Subsequence Problem is, given a sequence of *distinct* positive integers $A[1], \ldots, A[n]$, to find a longest increasing subsequence in it.

For example, consider the sequence $7, 3, 8, 2, 4, 6, 5$. A longest increasing subsequence is $3, 4, 6$, another longest increasing subsequence is $2, 4, 5$.

This question asks you to give an algorithm that solves the Longest Increasing Subsequence Problem using the dynamic programming technique.

a [**3**] Describe the array that you want to compute.

b [**3**] Give the initial values and the recurrence for computing the elements in the array.

c [**5**] Give the pseudo-code for computing the array.

d [**5**] Give the program for computing the optimal solution, using the array described in part **a)**.