Duration: **50 minutes**
Aids allowed: ***one single*-sided *hand*written** 8.5 × 11 **aid sheet**

Student number: ______________________
Last (Family) Name(s): ______________________
First (Given) Name(s): ______________________

Do **not** *turn this page until you have received the signal to start.*
In the meantime, please read instructions below carefully.

This term test consists of 3 questions. When you receive the signal to start, please make sure that your copy of the test is complete and fill in the identification section above.

Answer each question directly on the test paper, in the space provided, and use one of the "blank" pages for rough work. If you need more space for one of your solutions, use a "blank" page and *indicate clearly the part of your work that should be marked.*

Write up your solutions carefully! In particular, use notation and terminology correctly and explain what you are trying to do.

You will receive 20% of the points for any (sub)problem for which you write "I do not know how to answer this question." You will receive 10% if you leave a question blank. If instead you submit irrelevant or erroneous answers you will receive 0 points. You may receive partial credit for the work that is clearly "on the right track."

Marking Scheme

# 1:	_____/20
# 2:	_____/20
# 3:	_____/20
TOTAL:	_____/60

Good Luck!

Question 1 [20 marks]

Given a sorted array of *distinct* integers $A[1..n]$, you want to find out whether there is an index i for which $A[i] = i$. Design a decrease-and-conquer (special case of divide-and-conquer with a single subproblem solved recursively) algorithm that runs in $O(\log n)$ time.

(a) Describe your algorithm in plain English (at most 5 short sentences).

(b) Describe your algorithm in pseudocode.

(c) State a useful loop/recursion invariant (without proof). State the termination condition(s) and how it implies correctness.

(d) Prove that the running time of your algorithm is $O(\log n)$ (you may use Master's theorem). Make sure any notation you introduce is defined precisely.

This page is intentionally left (almost) blank.

Question 2 [20 MARKS]

Ternary Huffman. Trit Computing Inc. has developed a new computer architecture that operates on trits instead of bits. Regular bits can assume values 0 or 1 only, while trits can assume values 0, 1, or 2. In particular, Trit Computing Inc. produces solid state drives that store trits instead of regular bits. You, as an expert in data compression, have been hired to develop a modified Huffman algorithm to take advantage of their new storage technology. Trit Computing Inc. needs to compress sequences of characters from an alphabet of size n ($n \geq 3$, n is odd), where the characters occur with known frequencies $f_1, f_2, \ldots, f_n$. Your algorithm should encode each character with a variable-length prefix-free code over the values 0, 1, 2 so as to obtain maximum possible compression. Your algorithm should return a ternary tree representation of such code.

(a) Describe your algorithm in plain English (at most 5 short sentences).

(b) Describe your algorithm in pseudocode. You may use priority queues in your algorithm.

(c) Prove correctness of your algorithm. If your algorithm is greedy justify the greedy choice, state and prove optimal substructure property.

(d) State the running time of your algorithm. Justify it. State any assumptions on the implementation of abstract data types that are needed to achieve this running time.

This page is intentionally left (almost) blank.

Question 3 [20 MARKS]

You are given a string of n characters $s[1..n]$, which you believe to be a corrupted text document in which all punctuation has vanished (so that it looks something like "ireallylikedynamicprogrammingproblems..."). You wish to reconstruct the document using a dictionary, which is available in the form of a Boolean function dict($\cdot$): for any string w,

$$\text{dict}(w) = \begin{cases} \text{true}, & \text{if } w \text{ is a valid word} \\ \text{false}, & \text{otherwise.} \end{cases}$$

Give an efficient dynamic programming algorithm that determines whether the string $s[1..n]$ can be reconstructed as a sequence of valid words.

(a) Describe the semantic array.

(b) Describe the computational array. Don't forget the base case (if there is any).

(c) Justify why the above two arrays are equivalent.

(d) Write pseudocode to compute the *value (i.e., true or false)* of a solution.

(e) State the running time of your algorithm. Provide a brief justification. Assume that calls to dict($\cdot$) take unit time.