Duration: **50 minutes**
Aids Allowed: ***One single-sided handwritten 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 the instructions below *carefully*.

---

This term test consists of 3 questions on 8 pages (including this one), printed on both sides of the paper. *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 any 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*.

In your answers, you may use without proof any theorem or result covered in lectures, tutorials, problem sets, or assignments, as long as you give a clear statement of the result(s)/theorem(s) you are using. You must justify all other facts required for your solutions.

Write up your solutions carefully! In particular, use notation and terminology correctly and explain what you are trying to do—partial marks *will* be given for showing that you know the general structure of an answer, even if your solution is incomplete.

However, if you do not know how to approach a problem, remember that writing "I do not know how to approach this problem." (or a similar statement) earns you 20% of the marks for a solution (not writing this will still earn you 10%), but only if you leave the question *entirely blank* (or cross off everything you wrote to make it clear that it should not be marked).

Marking Guide

| | |
|---|---|
| N° 1: | _____/20 |
| N° 2: | _____/20 |
| N° 3: | _____/20 |
| TOTAL: | _____/60 |

*Good Luck!*

**Question 1.** Divide-and-Conquer [20 MARKS]

Write a divide-and-conquer algorithm that finds the maximum difference between any two elements of a given array of $n$ numbers (not necessarily distinct) *in* $O(n)$ *time*. For example, on input $A = [4.5, 10, -2, \pi, -7.115]$, your algorithm should return 17.115.

Justify briefly that your algorithm is correct and runs within the required time bound. (For your reference, the Master Theorem states that a recurrence of the form $T(n) = aT(n/b) + \Theta(n^d)$ has solution $\Theta(n^d)$ if $a < b^d$, $\Theta(n^d \log n)$ if $a = b^d$, and $\Theta(n^{\log_b a})$ if $a > b^d$.)

**Note:** For full marks, your answer *must* make use of the divide-and-conquer method. Partial marks will be given for an $O(n \log n)$ divide-and-conquer method.

*Use the space on this "blank" page for scratch work, or for any solution that did not fit elsewhere.*
***Clearly label each such solution with the appropriate question and part number.***

## Question 2. Greedy [20 MARKS]

Consider the following *Art Gallery Guarding* problem. We are given a set $\{p_1, p_2, \ldots, p_n\}$ of positive numbers that specify the positions of paintings in a long hallway in an art gallery. We want to position guards in the hallway to protect every painting, using as few guards as possible. Suppose that a guard at location $x$ can protect all paintings within 1 distance (i.e. in interval $[x-1, x+1]$), and any painting protected by at least one guard is considered protected.

We propose to solve this problem using the following greedy algorithm.

```
Sort the positions so p_1 ⩽ p_2 ⩽ ··· ⩽ p_n.
G ← ∅    # current set of guards' locations
g ← −∞    # position of rightmost guard in G = maximum number in G
for i ← n,...,1:    # start at the rightmost painting and move to the left
    if p_i − g > 1:    # p_i is unprotected by the guards currently in G
        # Place a guard 1 unit to the right of p_i.
        g ← p_i + 1
        G = G ∪ {g}
return G
```

(a) (2.5 marks) The above algorithm does not quite work. Provide a counterexample of painting positions where the above algorithm would use more guards than required. No justification is needed.

(b) (2.5 marks) The above algorithm can be fixed by changing a single line. What is the change needed to make the algorithm always find a solution with the minimum number of guards? For partial marks, you can propose any other greedy algorithm which always finds a solution with the minimum number of guards.

(c) (2.5 mark) What is the time complexity of your algorithm from part (b)?

(d) (10 marks) Prove that your fix to the algorithm (or any other greedy algorithm you proposed) in (b) is correct.

(e) (2.5 marks) The exhibition is then moved to another gallery where the art is put up on easels in a wide open space, i.e., each painting is placed at a 2D point $p_i = (x_i, y_i)$. Suppose we use a variant of your fixed greedy algorithm from part (b), where you sort the paintings by x-coordinate, check if a painting $p_i$ is greater than a unit Euclidean distance from the current guard (i.e., $\|p_i - g\| > 1$), and if so, place a new guard at $g \leftarrow p_i + (1, 0)$. Either prove that this algorithm will always return a solution with the minimum number of guards or provide a counterexample.

**Note:** The Euclidean distance is given by $\|(x, y) - (x', y')\| = \sqrt{(x - x')^2 + (y - y')^2}$.

*Use the space on this "blank" page for scratch work, or for any solution that did not fit elsewhere.*
***Clearly label each such solution with the appropriate question and part number.***

**Question 3.** Dynamic Programming [20 MARKS]

A new country Oddistan has a currency *oddollar* with coins of denomination 3, 5 and 7.

(a) (2.5 marks) Can you always purchase an item worth $n$ oddollars, where $n >= 5$, with exact change in coins. If so, explain why; otherwise, provide the smallest counterexample $n \geqslant 5$ (no justification is needed).

(b) (2.5 marks) For all $n$ that you can pay for exactly, is the combination of coins unique? If so, explain why; otherwise, provide a example $n$ and two ways to make up $n$ using the coin denominations.

(c) (15 marks) A new government wants to the change the coin denominations to 3, 11 and 13. Write a dynamic programming algorithm that returns (true/false) whether an exact change can be made for $n$ oddollars, given any integer $n > 0$. Clearly define the quantity that your dynamic programming algorithm computes, write a Bellman equation, briefly argue that your Bellman equation is correct, identify the initial conditions, and analyze the time and space complexity of your algorithm.

*Use the space on this "blank" page for scratch work, or for any solution that did not fit elsewhere.*
***Clearly label each such solution with the appropriate question and part number.***

*Use the space on this "blank" page for scratch work, or for any solution that did not fit elsewhere.*
***Clearly label each such solution with the appropriate question and part number.***