Note to Students: This file contains sample solutions to the term test together with the marking scheme and comments for each question. Please read the solutions and the marking schemes and comments carefully. Make sure that you understand why the solutions given here are correct, that you understand the mistakes that you made (if any), and that you understand *why* your mistakes were mistakes.

Remember that although you may not agree completely with the marking scheme given here it was followed the same way for all students. We will remark your test only if you clearly demonstrate that the marking scheme was not followed correctly.

For all remarking requests, please submit your request **in writing** directly to your instructor. For all other questions, please don't hesitate to ask your instructor during office hours or by e-mail.

Question 1. [10 marks]

The following problem is known as the **Art Gallery Guarding** problem. We are given a line L that represents a long hallway in an art gallery. We are also given a set $X = \{x_1, \dots, x_n\}$ of positive real numbers that specify the positions of paintings in this hallway. Suppose that a single guard can protect all the paintings within distance at most 1 of his or her position (on both sides). A solution to the problem uses as few guards as possible to guard all the paintings whose positions are represented by X.

Part (a) [5 marks]

Design a Greedy Algorithm to solve the art gallery guarding problem. Include a clear, concise, high-level English description of the main idea behind your algorithm.

Sample Solution:

Main idea: Start at the leftmost painting and move to the right. Place a guard whenever you are 1 unit to the right of the leftmost currently unguarded painting.

Algorithm:

```
lastGuard := $-\infty$
$S := \varnothing$
for $i := 1, \dots, n$ do:
    if $x_i >$ lastGuard $+ 1$ then
        lastGuard := $x_i + 1$
        $S = S \cup \{$lastGuard$\}$
    end if
end for
return $S$
```

Runtime: $\Theta(n)$

Marking Scheme:

- 2 marks: idea is probably correct but description is unclear enough that there is some ambiguity
- 2 marks: correct algorithm
- 1 mark: correct runtime computation

Error Codes:

Marker's Comments:

Part (b) [5 marks]

Prove the correctness of your algorithm.

Sample Solution:

We will prove by induction that for every $0 \leqslant j \leqslant n$, the set of guards $\mathcal{S}_j$ chosen by the greedy algorithm to guard all the paintings at the locations $\{x_1, \ldots, x_j\}$ is the same as the set of guards chosen by some optimal algorithm $\mathcal{O}$. And hence, the greedy algorithm is itself optimal.

Base Case: $j = 0$: This implies there are no paintings to guard. So, the algorithm chooses $\mathcal{S}_j := \varnothing$, which is clearly a subset of every optimal solution.

Ind. Hyp.: For some $j \geqslant 0$, assume that there is an optimal solution $\mathcal{O}$ such that $\mathcal{S}_j$ and $\mathcal{O}$ choose the same guards to guard the paintings at locations $\{x_1, \ldots, x_j\}$.

Ind. Step: At stage $j+1$, there are two possibilities: either,

Case 1: The guards in the optimal solution $\mathcal{O}$ already guard the painting at x_{j+1}; or,

Case 2: The guards in the optimal solution $\mathcal{O}$ do not guard the painting at x_{j+1}.

If Case 1 happens, then $\mathcal{S}_{j+1} = \mathcal{S}_j$ since the guards in $\mathcal{S}_j$ guard the painting at the location x_{j+1}, and hence, $\mathcal{O}$ and $\mathcal{S}_{j+1}$ agree on all the guards guarding the paintings at the locations $\{x_1, \ldots, x_{j+1}\}$.

If Case 2 happens, then there are two further sub-possibilities: either,

SubCase 1: $\mathcal{O}$ places a guard at location $x_{j+1} + 1$, in which case again $\mathcal{S}_{j+1}$ and $\mathcal{O}$ agree on all chosen guards guarding the paintings at locations $\{x_1, x_2, \ldots, x_{j+1}\}$ (note that if $\mathcal{O}$ places any guard that is not in $\mathcal{S}_j$ but is to the left of $x_{j+1} + 1$, then $\mathcal{O}$ is not an optimum solution since such a guard can be removed); or,

SubCase 2: $\mathcal{O}$ does not place a guard at location $x_{j+1} + 1$. Let y be the position of the leftmost guard in $\mathcal{O}$ that is not in $\mathcal{S}_j$; this guard must guard the painting at location x_{j+1}, and hence must be to the left of $x_{j+1} + 1$. Define

$$\mathcal{O}' := \mathcal{O} \setminus \{y\} \cup \{x_{j+1} + 1\}.$$

Since a guard at position $x_{j+1} + 1$ guards the painting at x_{j+1} and also every painting to the right of the painting at x_{j+1} that is guarded by the guard at position y (and possibly more), it follows that $\mathcal{O}'$ is also a solution.
Since $\mathcal{O}$ and $\mathcal{O}'$ have the same size, it follows that $\mathcal{O}'$ is also optimal.
Moreover, $\mathcal{S}_{j+1}$ and $\mathcal{O}'$ share the same chosen guards that guard the paintings at the locations $\{x_1, \ldots, x_{j+1}\}$, as desired.

Marking Scheme:

A. 2 marks: correct format/structure

B. 3 marks: correct idea (part marks for incorrect proof with some correct ideas)

Error Codes:

Marker's Comments:

Question 2. [15 MARKS]

Write an algorithm using dynamic programming to solve the following problem:

- **Input:** Positive integers a, b, c, d, n.
- **Question:** Are there nonnegative integers (w, x, y, z) such that $n = aw + bx + cy + dz$? If yes, find one such solution for (w, x, y, z).

Analyze the running time of your algorithm. Follow the 5-steps "dynamic programming paradigm" covered in class. In particular, describe carefully the recursive structure of the problem in order to justify the correctness of your recurrence.

SAMPLE SOLUTION:

Recursive Structure: If $n = aw + bx + cy + dz$ for some $w, x, y, z \in \mathbb{N}$, then either

- $w > 0$ and $n = a + a(w-1) + bx + cy + dz$, or
- $x > 0$ and $n = aw + b + b(x-1) + cy + dz$, or
- $y > 0$ and $n = aw + bx + c + c(y-1) + dz$, or
- $z > 0$ and $n = aw + bx + cy + d + d(z-1)$.

Array Definition: Let $m = \max\{a, b, c, d\}$. Then, for $i = -m, \dots, -1, 0, 1, \dots, n$, define $A[i]$ = True if $i > 0$ and $i = aw + bx + cy + dz$ for some nonnegative integers w, x, y, z (False otherwise).

Recurrence using Array: $A[-m] = \cdots = A[-1] = \texttt{False}$
$A[0] = \texttt{True}$ $(0 = a \cdot 0 + b \cdot 0 + c \cdot 0 + d \cdot 0)$
$A[i] = A[i-a] \vee A[i-b] \vee A[i-c] \vee A[i-d]$, for $i = 1, \dots, n$ (from argument above)

Bottom-up Algorithm:
function solveEquation(n):

```
Define array $A$
for $i := -m, \dots, -1$ do:
    $A[i] :=$ False
end for
$A[0] :=$ True
for $i := 1, \dots, n$ do:
    $A[i] := A[i-a] \vee A[i-b] \vee A[i-c] \vee A[i-d]$
end for
return $A[n]$
```

Runtime: $\Theta(n + m)$.

Finding a solution:
function findSolutionToEquation(A, n):

```
if $A[n]$ = False then
    print "no solution possible"
    return
set $i := n$
set $w := x := y := z := 0$
while $i > 0$ do:
    if $A[i-a]$ = True then
        $i := i - a$
```

```
            $w := w + 1$
        **else if** $A[i - b]$ = True **then**
            $i := i - b$
            $x := x + 1$
        **else if** $A[i - c]$ = True **then**
            $i := i - c$
            $y := y + 1$
        **else**
            $i := i - d$
            $w := w + 1$
        **end if**
    **return** $w, x, y, z$
```

Runtime: $\Theta(n)$.

MARKING SCHEME:

A. 4 marks: **Structure:** clear attempt to define an array indexed by subproblems, to give a recurrence for the array values (with justification), and to write an iterative algorithm based on the recurrence — even if these are done incorrectly

B. 4 marks: **Idea:** answer contains the argument that one of the box sizes must be used in any solution, so we can simply try each one in turn — even if this is poorly structured or written up

C. 7 marks: **Details:** correct array definition, correct recurrence with appropriate base cases and justification, correct algorithm (even if based on wrong recurrence), and correct runtime (even if algorithm is wrong)

ERROR CODES:

MARKER'S COMMENTS: