CISC-365*
Test #3
February 12, 2019

Student Number (Required) ______________________

Name (Optional)________________________________

This is a closed book test. You may not refer to any resources.

This is a 50 minute test.

Please write your answers in ink. Pencil answers will be marked, **but will not be re-marked under any circumstances.**

The test will be marked out of 50.

Question 1	/28
Question 2	/20
Question 3	/2
TOTAL	/50

Question 1 (28 marks)

Congratulations! Your international prestige as a problem-solver has earned you a new job – you now operate a guided-tour business in Balatronia.

Tourists sign up for 1-week (Short) or 2-week (Long) guided tours of the local mud pits during the summer season. There is a Short tour and a Long tour starting each week except the last week of the summer - in which there is only a Short tour. Each tour is worth a different amount of tip money, based on the wealth of the tourists. Your goal is to decide which tours to guide personally, without choosing any overlapping tours.

For example, suppose the summer season is 5 weeks long. The tours starting in each week might look like this. Tours are numbered according to the week in which they start.

	Week 1	Week 2	Week 3	Week 4	Week 5
1-week tours	$Short_1$ Value = 10	$Short_2$ Value = 7	$Short_3$ Value = 12	$Short_4$ Value = 4	$Short_5$ Value = 9
2-week tours	$Long_1$ Value = 20		$Long_3$ Value = 18		
		$Long_2$ Value = 22		$Long_4$ Value = 16	

One solution is to choose $Short_1, Long_2, Short_4, Short_5$ with a total value of 45

A better solution is to choose $Long_1, Short_3, Long_4$ with a total value of 48

In Week 1, you can guide either the 1-week tour ($Short_1$) or the 2-week tour ($Long_1$). In Week 2, you are either halfway through tour $Long_1$ or you can start guiding either of the tours that start in Week 2 (if you chose $Short_1$ in Week 1).

This question asks you to construct a Dynamic Programming solution to maximize your personal profit. Your solution must work on all instances, not just the example shown here.

(a) (5 marks) Explain how this problem satisfies the Principle of Optimality . Your explanation must be clear but a rigourous proof is not required.

(Hint: Suppose the optimal solution contains a particular tour X. What can you say about the chosen tours that precede X, and the chosen tours that follow X?)

This material was not covered in F2019 … but I have included the solution here in case you are interested!

Solution:

Based on the hint: if tour X is in the optimal solution, then the chosen tours that precede X must be a solution to the subproblem of choosing tours within the weeks before X begins. Similarly, the chosen tours that follow X must be a solution to the subproblem of choosing tours within the weeks after X ends. These must be optimal solutions to these subproblems because if they weren't we could replace them by something better, which would improve the overall optimal solution – which is not possible.

It is also possible to focus on the last tour in the optimal solution – it will be either $Short_n$ ***or*** $Long_{n-1}$***. Whichever it is, the other tours in the optimal solution must be an optimal solution within the weeks preceding the final tour in the optimal solution.***

We could also focus on the first tour in the optimal solution – it will be either $Short_1$ ***or*** $Long_1$ ***. Whichever it is, the other tours in the optimal solution must be an optimal solution within the weeks following the first tour in the optimal solution.***

Marking:

A solution similar to any of the above	5
A solution that shows understanding of the P. of O. without applying it successfully to this problem	3
For trying	1

(b) (8 marks) Give a recurrence relation for this problem.

Hint: Suppose the season is n weeks long. At the end of Week n, you will either be finishing $Short_n$ (and getting its value) or finishing $Long_{n-1}$ (and getting its value). Associate each of these possibilities with the appropriate subproblem. You may want to use "$P(k)$" to represent the maximum profit you can get in the first k weeks of the season.

Solution:

Defining P(k) as above and Val(T) to be the value of tour T, we can use

Recursive part:

for k $\geq$ 2

```
P(k) = max (  Val($Short_k$) + P(k-1),          # finish with a short tour
              Val($Long_{k-1}$) + P(k-2)        # finish with a long tour
           )
```

Base cases:

P(0) = 0

P(1) = Val($Short_1$)

Marking:

for a correct recursive part	5
for a partially correct recursive part	3
for trying	1
for a correct base	3
for a partially correct base	2
for trying	1

Students might omit the P(0) = 0 base case. That's ok but their recursive part has to be written in such a way that it never tries to recurse to P(0) … so if it refers to P(k-2), it must ensure that k > 2.

(c) (5 marks) Explain and justify the order in which you will compute solutions to subproblems. If you plan to use a table to store solutions to subproblems, this is the place to describe it.

Solution:

Since the recurrence relation only has one parameter, we can use a 1-dimensional array A to hold the solutions to the subproblems : A[i] will be used to store the value of P(i). The array should be indexed from 0 to n. The array is initialized with A[0] = 0, A[1] = P(1).

After that, the elements of the array are filled in ascending order. Each element's value depends on the two values immediately to its left. This order is chosen because it traverses the array in a natural manner and each element's value is computed as soon as the information needed is available.

It is also acceptable to manage the filling of the table using recursion (or a stack!) to keep track of the subproblems encountered. Each subproblem is encountered multiple times but solved only once. Due to the nature of this particular problem, the table will still be filled in from left to right!

Marking:

For any rational plan for the order of solving the subproblems	5
For an explanation with minor/significant/major errors	4/3/2
For trying	1

(d) (5 marks) Explain how you will determine the details of the optimal solution.

Solution:

When we know the optimal final value, we can look at the two values immediately to its left in A to determine which of those options led to the optimal answer. This tells us whether we ended with a Short or Long tour. From whichever element of A led to the final answer, we repeat this process to determine the tour we choose before the last one … and so on back to the start of the summer.

Alternatively, the table could have been defined to also contain information regarding the elements of the optimal solution. In that case, the extraction of this information would be based on how it was stored.

Marking:

For a reasonably clear explanation of how to get the information	5
For an explanation with minor/significant/major errors	4/3/2
For trying	1

(e) (5 marks) What is the complexity of your algorithm? (Use n to represent the number of weeks in the summer season)

Solution:

Each element of A is computed in constant time, so filling A takes O(n) time.

Each step of the "trace back" is determined in constant time and there are at most n steps, so finding the details of the optimal solution takes O(n) time.

Thus the entire algorithm takes O(n) time.

Marking:

For a correct analysis of their version of the algorithm	5
For an explanation with minor/significant/major errors	4/3/2
For trying	1

QUESTION 2 (20 Marks)

You and your worst enemy are playing a game. Between you are three piles of coins, containing n_1, n_2 and n_3 coins respectively. You take turns removing coins according to this rule: on your turn you must remove a positive number of coins from any one of the piles (ie you must take at least 1 coin). **You win the game if you take the very last coin.**

Each possible game situation is described by the sizes of the piles such as (4,7,2) or (2019,3,12)

If a single move can get from (a, b, c) to (d, e, f) we call (d, e, f) a ***child*** of (a, b, c). For example, we can get from $(8, 7, 5)$ to $(8, 4, 5)$ by removing 3 coins from the centre pile so $(8, 4, 5)$ is a child of $(8, 7, 5)$

We can label a game situation "W" if the player who takes the next turn can be sure of winning, and "L" if they can't. For example $(0, 0, 5)$ is a "W" situation – the player can take the whole third pile, but $(1, 1, 0)$ is an "L" - the player must take 1 coin, then the other player takes the last coin and wins.

In general, a situation is "W" if ***any*** of its children is labelled "L", and a situation is "L" if ***all*** of its children are labelled "W"

Create a recurrence relation to determine if situation (n_1, n_2, n_3) is a "W" or "L"

(Write your answer on the next page)

(a) (10 marks) Recursive part:

Solution:

I will use G(a,b,c) to represent the label of the game when the three piles have sizes a, b, and c.

G(a,b,c) = "W" if G(a,b,x) = "L" for any x in the range [0..c-1] or
G(a,x,c) = "L" for any x in the range [0..b-1] or
G(x,b,c) = "L" for any x in the range [0..a-1]

= "L" if G(a,b,x) = "W" for all x in the range [0..c-1] and
G(a,x,c) = "W" for all x in the range [0..b-1] and
G(x,b,c) = "W" for all x in the range [0..a-1]

The two cases given cover all of the possibilities, so it is not actually necessary to specify both. For example

G(a,b,c) = "W" if G(a,b,x) = "L" for any x in the range [0..c-1] or
G(a,x,c) = "L" for any x in the range [0..b-1] or
G(x,b,c) = "L" for any x in the range [0..a-1]

= "L" otherwise

is perfectly acceptable

Marking:	**- for a correct solution**	**10**
	- for a solution that is mostly correct with a minor error such as making the ranges start at 1 instead of 0	**8**
	- for a solution with a major error such as leaving out one of the sets of subproblems	**6**
	- for a solution that looks like a recurrence relation for the game but is seriously wrong	**4**
	- for a solution that shows limited understanding of what is required	**2**
	- for trying	**1**

(b) (10 marks) Base case(s):

Solution:

The following is sufficient:

$$G(0,0,0) = \text{“L”}$$

but students may include others such as

$$G(0,1,1) = G(1,0,1) = G(1,1,0) = \text{“L”}$$

Students may use other sets of base cases such as

$$G(0,0,x) = G(0,x,0) = G(x,0,0) = \text{“W”} \quad \text{for all } x > 0$$
$$G(0,x,x) = G(x,0,x) = G(x,x,0) = \text{“L”} \quad \text{for all } x \geq 0$$

Another possible answer is (see note below)

$$G(x,y,y) = G(y,x,y) = G(y,y,x) = \text{“W”} \quad \text{for all } x > 0 \text{ and } y \geq 0$$

The important thing is to have a set of base cases such that

- **every possible sequence of moves (eventually) reaches one of the base cases**
- **situations with one or more empty pile are covered**

Note: Students might not include a base case for (0,0,0) since that actually signifies the end of the game. That's ok as long as they "cover" all the states that lead to (0,0,0) so the recursion can't end up at (0,0,0) and not have a resolution. The final answer shown above is an example of such a set of base cases.

Marking:

pretty much the same as Part (a). As noted above, it is important that every sequence of moves in the game ends up in a base case.

QUESTION 3 (2 Marks)

True or False:

It was just a semi-frivolous T/F question. The correct answer was "FALSE"