NOTE: This file contains sample solutions to the quiz 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 quiz 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.

GENERAL MARKING SCHEME:

- ***A****: All Correct*, except maybe for very few minor errors.
- ***B****: Mostly Correct*, but with a few serious errors, or many small errors.
- ***C****: Mostly Incorrect*, but with a few important elements, or many small elements, done correctly.
- ***10%****: Completely Blank*, or clearly crossed out.
- ***D****: All Incorrect*, except maybe for very few minor elements done correctly.

MARKER'S COMMENTS: *(None)*

INSTRUCTOR'S COMMENTS: As I pointed out on the course forum, I realize that this quiz was too long (though not too difficult). I will make adjustments to account for this.

1. Consider the following "Longest Increasing Sublist" problem.

Input: A list of integers $L = [a_1, a_2, \ldots, a_n]$.

Output: A sublist $L' = [a_{i_1}, a_{i_2}, \ldots, a_{i_k}]$ such that $1 \leqslant i_1 < i_2 < \cdots < i_k \leqslant n$ and $a_{i_1} < a_{i_2} < \cdots < a_{i_k}$ and k is maximum.

For example, if $L = [4, 1, 7, 3, 10, 2, 5, 9]$, then $L_1 = [1, 3, 5, 9]$ and $L_2 = [1, 2, 5, 9]$ are two optimal solutions, but $[1, 2, 3, 4]$ is not a solution (it takes integers from L out of order), $[1, 7, 3, 10]$ is not a solution (it is not increasing), and $[4, 7, 10]$ is not an optimal solution (it is not as long as possible).

Give a dynamic programming algorithm to solve the Longest Increasing Sublist problem. (HINTS: Any optimal solution for input $[a_1, a_2, \ldots, a_n]$ either contains a_n, or it does not. Consider sub-problems whose solutions have last element a_k, for various values of k.)

Step 0: *Describe the recursive structure of sub-problems.*
(See the hint.)

Step 1: *Define an array that stores optimal values for arbitrary sub-problems.*
Let $M[k]$ represent the length of a longest increasing sublist that ends with a_k, for $k = 1, 2, \ldots, n$.

Step 2: *Give a recurrence relation for the array values.*
$M[k] = \max\{M[i] + 1 : 0 < i < k \wedge a_i < a_k\}$, for $k = 1, 2, \ldots, n$.

Step 3: *Write a bottom-up algorithm to compute the array values, following the recurrence.*

```
for k in [1,2,...,n]:
    M[k] := 1
    for i in [1,2,...,k-1]:
        if a_i < a_k and M[i] + 1 > M[k]:
            M[k] := M[i] + 1
```

Step 4: *Use the computed values to reconstruct an optimal solution.*

Use a second array $N[k]$ to store the index of the second-last element in the longest sub-list that ends with a_k.

```
# Complete algorithm.
for k in [1,2,...,n]:
    M[k] := 1
    N[k] := k
    for i in [1,2,...,k-1]:
        if a_i < a_k and M[i] + 1 > M[k]:
            M[k] := M[i] + 1
            N[k] := i

# Figure out the last element in the longest increasing sub-list.
b := 1
for k in [2,3,...,n]:
    if M[k] > M[b]:  b = k

# Generate the sub-list, working backwards.
S := [b]
while N[b] != b:
    S := N[b] + S
    b := N[b]
```