MAT2440/D641 **Exam #3**
Instructor: Suman Ganguli **Wednesday, November 28** **Name:** _____

Question	Points	Score
1	10	
2	10	
3	10	
4	10	
5	10	
Total:	50	

1. (10 points) (a) Consider a function "`sort`" which takes as input a list of 5 integers and returns the list sorted in ascending order. For example:

$$\texttt{sort}(9, 4, 0, 5, -1) = (-1, 0, 4, 5, 9)$$

i. What is the domain of `sort`?

Solution:

$$\mathbb{Z} \times \mathbb{Z} \times \mathbb{Z} \times \mathbb{Z} \times \mathbb{Z} = \mathbb{Z}^5$$

ii. Show that `sort` is *not* one-to-one.

Solution: `sort` is clearly not one-to-one, since there are many pairs of inputs in the domain $\mathbb{Z}^5$ which get mapped to the same output by `sort`. Just take any two lists of the same 5 integers listed in different orders. For example:

$$\texttt{sort}(9, 4, 0, 5, -1) = \texttt{sort}(4, 9, 0, 5, -1) = (-1, 0, 4, 5, 9)$$

(b) Recall the floor function, which assigns to the real number x the largest integer that is less than or equal to x, and is denoted by $\lfloor x \rfloor$. For example:

$$\lfloor 0.5 \rfloor = 0$$
$$\lfloor -2.1 \rfloor = -3$$
$$\lfloor 8.9 \rfloor = 8$$

i. What is the domain of the floor function? What is the range of the floor function?

Solution: The domain is $\mathbb{R}$ (the set of real numbers) and the range is $\mathbb{Z} = \{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\}$ (the set of integers).

ii. Show that the floor function is *not* one-to-one.

Solution: Floor is clearly not one-to-one, since there are many pairs of real numbers which have the same floor. For example,

$$\lfloor 0.5 \rfloor = \lfloor 0.1 \rfloor = 0$$

2. (10 points) Here is pseudocode which implements binary search:

procedure `binary-search` (x : integer, $a_1, a_2, \ldots a_n$: increasing integers)
  $i := 1$ (the left endpoint of the search interval)
  $j := n$ (the right endpoint of the search interval)
  while $(i < j)$:
  $m := \left\lfloor \frac{i+j}{2} \right\rfloor$
  if $(x > a_m)$ **then**: $i := m + 1$
  else: $j := m$
  if $(x = a_i)$ **then**: location := i
  else: location := 0
  return location

Fill in the steps used by this implementation of binary search to find the location of $x = 38$ in the list

$$a_1 = 17, a_2 = 22, a_3 = 25, a_4 = 38, a_5 = 40, a_6 = 42, a_7 = 46, a_8 = 54, a_9 = 59, a_{10} = 61$$

Solution:

- Step 1: Initially $i = 1, j = 10$ so search space is the entire list

  $$a_1 = 17, a_2 = 22, a_3 = 25, a_4 = 38, a_5 = 40, a_6 = 42, a_7 = 46, a_8 = 54, a_9 = 59, a_{10} = 61$$

- Step 2: $m = \left\lfloor \frac{11}{2} \right\rfloor = \lfloor 5.5 \rfloor = 5$ and so $a_m = a_5 = 40$

  Since $x = 38 \leqslant a_m = 40$, the algorithm resets the right endpoint j to $m = 5$ (i.e., it cuts the search interval down to the left half of the initial list):

  $i = 1, j = 5$

  so the new search interval is: $a_1 = 17, a_2 = 22, a_3 = 25, a_4 = 38, a_5 = 40$

- Step 3:

  $m = \left\lfloor \frac{6}{2} \right\rfloor = \lfloor 3 \rfloor = 3$ and so $a_m = a_3 = 25$

  Since $x = 38 > a_m = 25$, the algorithm resets the left endpoint i to $m + 1 = 4$ (i.e., it cuts the search interval down to the right half of the list from Step 2):

  $i = 4, j = 5$

  so the new search interval is: $a_4 = 38, a_5 = 40$

- Step 4: $m = \left\lfloor \frac{9}{2} \right\rfloor = \lfloor 4.5 \rfloor = 4$ and so $a_m = a_4 = 38$

  Since $x = 38 \leqslant a_m = 38$, the algorithm resets the right endpoint j to $m = 4$:

  $i = 4, j = 4$

  so the search interval is: $a_4 = 38$

- Step 5: Since $i = j$ (i.e., the algorithm has reduced the search interval to a single element), the algorithm escapes the "while" loop, and checks whether $x = a_i$. Since $x = 38 = a_4 = 38$, the algorithm returns the variable "location" with value $i = 4$.

3. (10 points) Suppose you are given two finite sets $A = \{a_1, a_2, \ldots, a_n\}$ and $B = \{b_1, b_2, \ldots, b_m\}$. Write an algorithm in pseudocode which prints out all elements of the Cartesian product $A \times B = \{(a, b) \mid a \in A \text{ and } b \in B\}$. (Hint: Use nested for-loops.)

Solution: We need to loop thru each of the elements of A, and for each $a \in A$ loop thru each of the elements of B, in order to generate all such ordered pairs:

procedure `print-Cartesian-product` $(A = \{a_1, a_2, \ldots a_n\}, B = \{b_1, b_2, \ldots, b_m\})$

  for $i = 1$ to n:

    for $j = 1$ to m:

      print "(a_i, b_j)"

4. (10 points) Find the terms in the sequence defined by the following recurrence relations and initial conditions:

(a)

$$a_n = -3a_{n-1}, a_0 = 1$$

What is a formula for a_n, as a function of n?

- $a_n =$

Solution:

$a_1 = -3a_0 = -3(1) = -3$
$a_2 = -3a_1 = -3(-3) = 9$
$a_3 = -3a_2 = -3(9) = -27$
$a_4 = -3a_3 = -3(-27) = 81$
It should be clear that $a_n = (-3)^n$

(b)

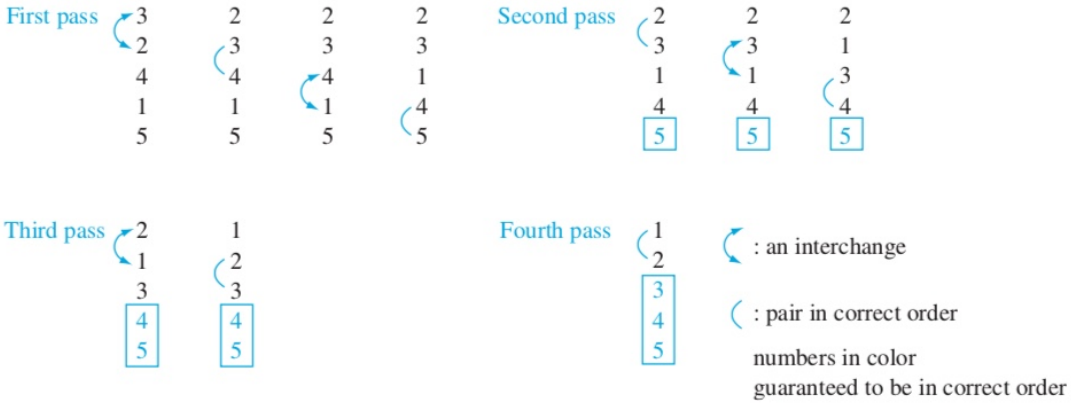
$$a_n = a_{n-1} + a_{n-2}$$
$$a_0 = 4, a_1 = 7$$

Solution:

$a_2 = a_1 + a_0 = 4 + 7 = 11$
$a_3 = a_2 + a_1 = 7 + 11 = 18$
$a_4 = a_3 + a_2 = 11 + 18 = 29$
$a_5 = a_4 + a_3 = 18 + 29 = 47$

Note that this is the same recurrence relation as the Fibonacci sequence, but with different initial conditions. This sequence is called the Lucas sequence. See `https://en.wikipedia.org/wiki/Lucas_number` and `https://en.wikipedia.org/wiki/Fibonacci_sequence`.

5. (10 points) Shown is an example of using bubble sort to sort a list of integers:



(a) Briefly describe the bubble sort algorithm. You can use the example as a guide, but you should **describe how the algorithm works in general**, on any given list of numbers.

Solution: Bubble sort works by doing multiple "passes" through the list. On each pass the algorithm compares adjacent elements in the list and interchanges their positions if they are out of order. On the first pass, the algorithm goes through the entire list doing such interchanges. This ensures that the largest element "sinks" to the bottom of the list, i.e., to the n-th position. Thus, on the second pass, the algorithm only needs to go through the first $n-1$ elements; this sends the 2nd largest element in the list to the $(n-1)$-st position. Continuing in this way, the algorithm will require $n-1$ passes to sort the list (on an input list of length n).

(b) Use bubble sort to put the following list into alphabetical order, showing the lists obtained at each step of the algorithm (as in the example above).
Note that since there are 4 elements in the list, bubble sort requires $4-1=3$ passes:

Solution:

First pass:

NY	NJ	NJ
NJ	NY	NY
PA	PA	CT
CT	CT	PA

Second pass:

NJ	NJ
NY	CT
CT	NY
PA	PA

Third pass:

NJ	CT
CT	NJ
NY	NY
PA	PA

Sorted list:

CT
NJ
NY
PA