MAT2440/D648 **Exam #3**
Instructor: Suman Ganguli **Due: Thursday, May 18** **Name:** ____________

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

1. (10 points) Consider a function "`sort`" which takes as input a list of 5 integers (i.e., input $(a_0, a_1, a_2, a_3, a_4)$ where each $a_i \in \mathbb{Z}$), and returns the list sorted in ascending order. For example:

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

(a) What is the domain of `sort`? Express the domain as a Cartesian product.

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

(b) Show that `sort` is *not* a one-to-one function.

Solution: `sort` is clearly not one-to-one, since there are many pairs of inputs in the domain which get sorted to the same output. Just take any two lists of the same integers listed in different orders. For example:

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

2. (10 points) We analyzed that the worst-case time complexity of linear search is $O(n)$ while the time complexity of binary search is $O(\log n)$.

(a) What does the variable n represent here?

Solution:
n is the size of the input array of items to be sorted: $(a_1, \ldots, a_n)$

(b) Briefly explain what aspect of the binary search algorithm makes its time complexity $O(\log n)$. (It may be helpful to do #2 before answering this question–included on the next page is the pseudocode for binary search.)

Solution:
The running time of binary search is $O(\log n)$ because the algorithm proceeds by dividing the "search space" in half repeatedly. Thus, if $n \approx 2^k$ for some integer k, then the algorithm requires $O(\log n)$ operations to narrow down the search space to a single element.

(c) Based on their big-O estimates, which of these search algorithms is preferable to use for large values of n? Why?

Solution:
Given that $\log n$ grows much more slowly than n as gets large, it is preferable to use binary search instead of linear search. We should note however that binary search requires that the list be sorted, which adds overhead to the running time.

3. (10 points) Here is pseudocode which implements binary search:

```
procedure binary-search ($x$ : integer, $a_1, a_2, \dots a_n$: 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 $a_m = a_4 = 38$

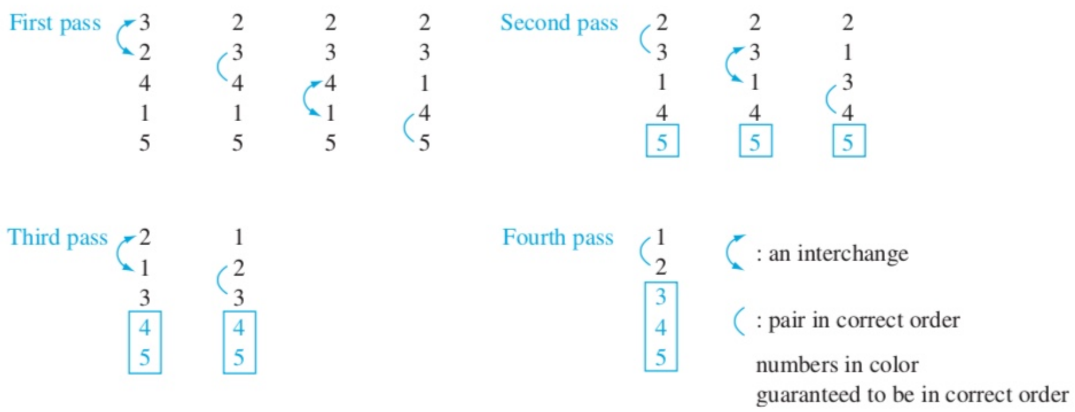
  Since $x = 38 \leqslant a_m = 38$, the algorithm resets the right endpoint j to $m = 4$:

  $i = 4, j = 4$

  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 $i = 4$.

4. (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 and the pseudocode (see the textbook or the slides) as a guide, but you should **describe how the algorithm works in general** (in your own words), on any given list of numbers. (Hint: Describe how many "passes" are made through the list, and what the algorithm does on each pass.)

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) **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

5. (10 points) Review the proof of the following theorem by mathematical induction (as presented in class and in the textbook, as Example 1 in Section 5.1):

Theorem: For any positive integer n,

$$1 + 2 + 3 + \ldots + n = \frac{n(n+1)}{2}$$

Fill in the steps in the proof of this theorem:

Proof (by induction):For any given positive integer n, we will use $P(n)$ to represent the proposition:

$$P(n) : 1 + 2 + 3 + \ldots + n = \frac{n(n+1)}{2}$$

Thus, we need to prove that $P(n)$ is true for $n = 1, 2, 3, ...$, i.e., we need to prove:

$$(\forall n \in \mathbb{N})P(n)$$

For a proof by mathematical induction, we must prove the base case (namely, that $P(1)$ is true), and we must prove the inductive step, i.e., that the conditional statement

$$P(k) \longrightarrow P(k+1)$$

is true, for any given $k \in\in \mathbb{N}$.

(a) **Base case:** Show that the base case $P(1)$ is true:

Solution: $P(1)$ is the equation $1 = \frac{1(1+1)}{2} = \frac{2}{2}$ which is clearly true.

(b) **Inductive step:** In order to provide a direct proof of the conditional $P(k) \longrightarrow P(k+1)$, we start by assuming $P(k)$ is true, i.e., we assume

$$1 + 2 + 3 + \ldots + k = \frac{k(k+1)}{2}$$

Now use this assumption to show that then $P(k+1)$ is true.
(Hint: note that the the proposition $P(k+1)$ is the equation:

$$1 + 2 + 3 + \ldots + k + (k+1) = \frac{(k+1)((k+1)+1)}{2}$$

Start with the LHS of this equation, and show that it is equal to the RHS, using the assumption/equation $P(k)$!)

Solution: We assume the "inductive hypothesis" $P(k) : 1 + 2 + 3 + \ldots + k = \frac{k(k+1)}{2}$. Then:

$$\begin{aligned}
1 + 2 + 3 + \ldots + k + (k+1) &= \frac{k(k+1)}{2} + (k+1) \\
&= \frac{k(k+1)}{2} + \frac{2(k+1)}{2} \\
&= \frac{k^2 + k + 2k + 2}{2} \\
&= \frac{k^2 + 3k + 2}{2} = \frac{(k+1)(k+2)}{2}
\end{aligned}$$

which is $P(k+1)$. Thus, we have showed that $P(k)$ implies $P(k+1)$.