There are 7 problems on the exam. The first and last are mandatory, and you may eliminate any one of problems 2 – 6 by drawing an X through it. Problem 1 is worth 10 points, and all other problems are worth 18. Please write in pen if possible. If you need more room, use the back of the sheet and tell me this on the front sheet of that problem.

**Problem 1 (10 pts -- Mandatory).** (True/False) Write True or False to the LEFT of the statement.

- False 1. The assignment double x = (double) 4; is an example of a narrowing conversion.
- False 2. In the boolean expression ( A && B ), if the expression A evaluates to true, the second expression B will not be evaluated.
- False 3. The break statement quits the current iteration of a loop and starts the next one.
- True 4. Local variables in a class can never be defined as public.
- True 5. Generic classes in Java can only be instantiated by reference types.
- True 6. From the point of view of \( \Theta(\cdot) \) complexity, the worst-case complexity of inserting a new element into an array is the same whether it is ordered or unordered.
- True 7. If \( f(n) = \Omega(g(n)) \) then it must be the case that \( f(n) = \Theta(g(n)) \).
- True 8. The average case complexity of an algorithm can never be worse (i.e., take longer) than the worst-case complexity.
- True 9. The scope of a member of a class is the same whether it is declared static or not static.
- True 10. Strings in Java are immutable, that is, you cannot change the characters in a String once you have created it.
Problem 2. (a) Perform Quicksort on the following array, according to the algorithm presented in lecture. If you get to a sub-problem of size 2, you may leave it as is (if they are in the right order) or exchange the two (if they are not in the right order). Show the array after every exchange and underline each pivot value.

5 1 6 4 9 3 2 7

(b) In general, what is the best possible choice of a pivot during the Partition step, and what is the time complexity that will result if this best choice happens every time?
(c) In general, what is the worst-possible choice of a pivot during the Partition step, and what is the time complexity that will result if this worst choice happens every time?
(d) Is it possible to design a version of Quicksort that avoids the worst case by choosing a “good” pivot value for all inputs? Answer “yes” or “no” and then explain in a short sentence.

(A) (Red indicates that the number is known to be in its final position—either the pivot was swapped or there is a sub-problem of size 1.)

5 1 6 4 9 3 2 7
5 1 2 4 9 3 6 7
5 1 2 4 3 9 6 7
3 1 2 4 5 9 6 7
2 1 3 4 5 9 6 7
1 2 3 4 5 9 6 7
1 2 3 4 5 7 6 9
1 2 3 4 5 6 7 9

(B) The best possible choice of a pivot value during the Partition step is a median (middle) value among the values of the array to be partitioned; if this happens every time, the time complexity will be \( \Theta(N \log(N)) \).

(C) The worst possible choice of a pivot value during the Partition step is an extreme (either maximal or minimal) value among the values of the array to be partitioned; if this happens every time, the time complexity will be \( \Theta(N^2) \).

(D) No! Since you can’t predict where the worst values might show up, no particular location (e.g., leftmost, center, rightmost) is better than any other in terms of the worst case. If you do anything “fancy” like try to find the median, then you will essentially have to sort the list before you partition it, which is silly, because sorting is what you are trying to do in the first place.
Problem 3.

(A) Complete the following method, which reverses a list of arbitrary size, under the assumption that the ONLY way you may move an element is to swap it with another element, i.e., you may not declare a temporary variable to store elements when you move them. You may declare variables to hold array indices however.

```java
// Reverse the integer array A
void reverse( int[] A ) {
    for(int i = 0; i < (A.length/2); ++i) {
        swap( A, i, (A.length-1)-i );
    }

    // OR:
    int L = 0; int R = A.length-1;
    while( L < R ) {
        swap(A, L, R);
        ++L; --R;
    }
}
void swap(int[] A, int i, int j) {
    int temp = A[i];
    A[i] = A[j];
    A[j] = temp;
}
```

(B) Suppose that we define the time complexity \(f(N)\) of your algorithm as the number of calls to `swap(...)` to reverse an array of size \(N\). Give the worst-case time complexity of your algorithm in terms of \(\Theta(...\), where the expression .... inside the parentheses is as simple as possible.

\[ \Theta(N) \]

(C) Under the same assumptions as (B), give the average-case complexity of your algorithm. Is this the same or different than your answer for (B)? Answer “same” or “different” and explain in a short sentence.

Same! The algorithm does the same steps regardless of the values in the array, so all cases (best, average, worst) are the same.
**Problem 4.** Suppose you have a stack for integers with the normal `push()` and `pop()` operations.

Now suppose you have a sequence of `push(...)` and `pop()` operations in the following form:

```java
// 0 or more statements of the form System.out.print( pop() + " " );
push( 0 );
// 0 or more statements of the form System.out.print( pop() + " " );
push( 1 );
// 0 or more statements of the form System.out.print( pop() + " " );
push( 2 );
  etc.
// 0 or more statements of the form System.out.print( pop() + " " );
push( 9 );
// 0 or more statements of the form System.out.print( pop() + " " );
```

That is, you push the digits 0 ... 9 in order, and then before, in between, and after these pushes, you do some number of pops (possibly 0).

Which of the following sequences could possibly be printed out by such a sequence (there may be more than one)? Write “possible” or “not possible” next to each one.

(A) 4 6 8 7 5 3 2 9 0 1  **NOT Possible!**

(B) 8 7 6 5 4 3 2 1 0 9  **Possible**

(C) 4 3 2 1 0 5 6 7 8 9  **Possible**

(D) 9 8 7 6 5 4 3 2 1 0  **Possible**
Problem 5. For each function $f$ from the following list of functions, determine the simplest function $g$ that makes the equality $f(N) = \Theta(g(N))$ true. Represent your answer as an equality (e.g., $p(N) = \Theta(N^2)$). The first one is done for you as a model for the rest of your answers.

\[ a(N) = 2^N + N^2 - 5 \quad \quad a(N) = \Theta(2^N) \]

\[ b(N) = \log(N^3) + \log(2N) \quad \quad b(N) = \Theta(\log(N)) \]

\[ c(N) = 8N + 5\sqrt{N} \quad \quad c(N) = \Theta(N) \]

\[ d(N) = 12N + N\sqrt{N} + N\log(N) \quad \quad d(N) = \Theta(N^{1.5}) \]

\[ e(N) = \log^3(N) + \log(N) + 5 \quad \quad e(N) = \Theta(\log^3(N)) \]

\[ f(N) = 50 + \frac{1}{N} \quad \quad f(N) = \Theta(1) \]
Problem 6. This is a question about Java and has two parts.

(A) What is the output of the following lines of Java code?

```java
int a = 9;
double b = a / 2.0;
int c = a / 2;
double d = a / 2;
String e = c + "c";
a = a % 6;

System.out.println("a = " + a); // a = 3
System.out.println("b = " + b); // b = 4.5
System.out.println("c = " + c); // c = 4
System.out.println("d = " + d); // d = 4.0
System.out.println("e = " + e); // e = 4c
```

(B) Consider the program shown below.

```java
public class Problem3 {
    public static void main(String[] args) {
        int g = 1;
        if (g > 0) {
            int h = foo(g);
        }
        System.out.println(______); // blank for part a
    }

    public static int foo(int r) {
        int s;
        for (s = 0; s < 5; s++) {
            System.out.println("bar");
        }
        return ______; // blank for part b
    }
}
```

a. Which of the variables (g, h, r, s) could be used in the first blank?

   g

b. Which of the variables (g, h, r, s) could be used in the second blank?

   r, s
Problem 7. (MANDATORY) For the following algorithms, for the first set, state the worst-case and average-case time (in terms of $\Theta$) for performing the indicating operation. For the second set, simply give the $\Theta(...N...)$ estimate for the indicated situation. The number of items stored in each data structure is $N$. Assume for each row that you are using the algorithms we discussed in class.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Worst Case $\Theta(...)$</th>
<th>Average Case $\Theta(...)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Find an element in an unordered array.</td>
<td>$\Theta(N)$</td>
<td>$\Theta(N)$</td>
</tr>
<tr>
<td>Delete an element from an ordered array.</td>
<td>$\Theta(N)$</td>
<td>$\Theta(N)$</td>
</tr>
<tr>
<td>Dequeue a key from a queue implemented as a ring buffer with resizing</td>
<td>$\Theta(1)$</td>
<td>$\Theta(1)$</td>
</tr>
<tr>
<td>Enqueue a key into a queue implemented as a ring buffer with resizing</td>
<td>$\Theta(N)$</td>
<td>$\Theta(N)$</td>
</tr>
<tr>
<td>Find an element in an ordered array.</td>
<td>$\Theta(\log(N))$</td>
<td>$\Theta(\log(N))$</td>
</tr>
<tr>
<td>Partition (as in Quicksort) a array of size $N$.</td>
<td>$\Theta(N)$</td>
<td>$\Theta(N)$</td>
</tr>
<tr>
<td>Merge two ordered lists, each of size $N/2$.</td>
<td>$\Theta(N)$</td>
<td>$\Theta(N)$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Algorithm and Data Input</th>
<th>$\Theta(...)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sort an already-sorted list using Selection Sort</td>
<td>$\Theta(N^2)$</td>
</tr>
<tr>
<td>Sort an already-sorted list using Insertion Sort</td>
<td>$\Theta(N)$</td>
</tr>
<tr>
<td>Sort an already-sorted list using Mergesort</td>
<td>$\Theta(N \log(N))$</td>
</tr>
<tr>
<td>Sort an already-sorted list using Quicksort</td>
<td>$\Theta(N^2)$</td>
</tr>
</tbody>
</table>