Name:

CS 112—Midterm

Summer 2, 2017

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.

T 1. The assignment double x = (char) 65 is an example of a narrowing conversion.

F 2. In the boolean expression (A && B), if the expression A evaluates to true, the second expression B will not be evaluated.

F 3. The break statement quits the current iteration of a loop and starts the next one.

T 4. Local variables in a method can never be defined as public.

F 5. The advantage of generic classes in Java is that you can put elements of different types (e.g., doubles and Strings) into the same array.

T 6. From the point of view of O(...) complexity, the worst-case complexity of inserting a new element into an array is the same whether it is ordered or unordered.

T 7. The average case complexity of an algorithm can never be worse (i.e., take longer) than the worst-case complexity.

T 8. The scope of a member of a class is the same whether it is declared static or not static.

T 9. Strings in Java are immutable, that is, you can not change the characters in a String once you have created it.

X 10. I have put my name on my exam.

Problem 2. (A) Perform **Mergesort** on the following array, according to the algorithm presented in lecture. (You may simply copy the values below, as if using a 2D array, as shown in class.)

	# MOVES	# COMPS
5 1 6 4 9 3 2 7		
1 5 4 6 3 9 2 7	8	1+1+1+1 = 4
1 4 5 6 2 3 7 9	8	3+3 = 6
1 2 3 4 5 6 7 9	8	6 = 6
	+ 24	+ 16

(B) State PRECISELY how many times you had to MOVE an integer during the sort, and give the O(…) estimate of this values.

24 $O(N \log N)$

(C) State PRECISELY how many times you had to COMPARE one integer in the array with another during the sort, and give the O(…) estimate of this value.

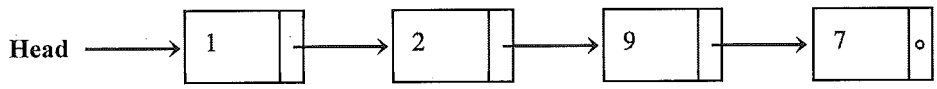
16 $O(N \log N)$

(D) How much extra memory is necessary for Mergesort when you want to sort an array of length N?

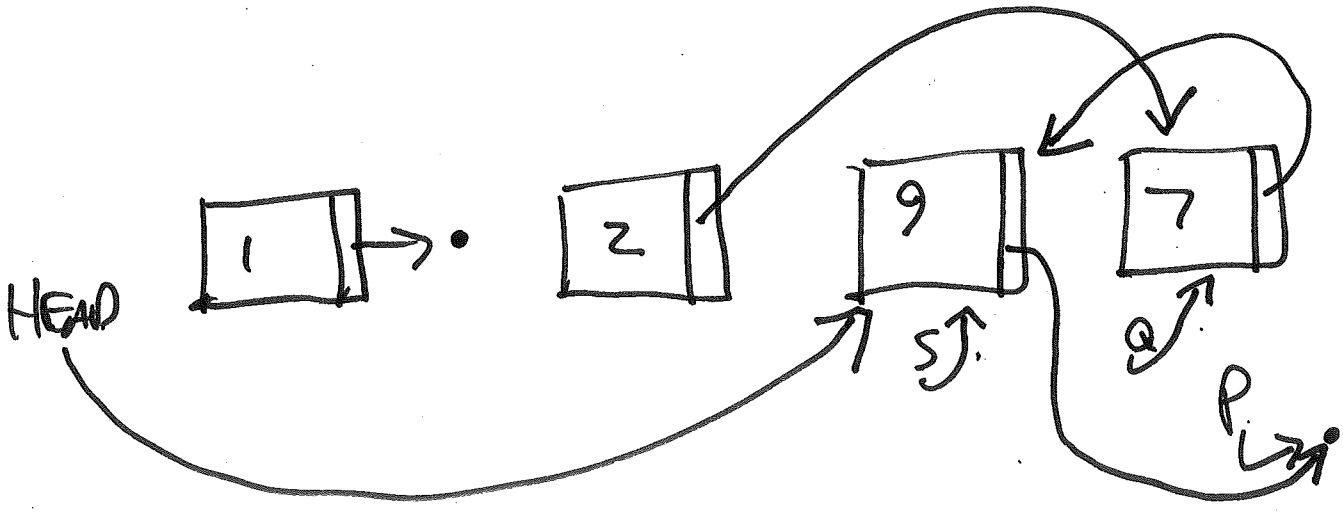
N or $O(N)$

Problem 3.

Run the following code on the linked list as shown, and draw the list that would result, and show what all of the pointers point to at the end of the algorithm.




```
Node p = head.next;
Node q = head;
Node s = null;

while ( p != null ) {
    s = q;
    q = p;
    p = p.next;
    q.next = s;
    s.next = p;
}

head.next = null;
head = s;
```



Problem 4. Suppose you have a **MaxQueue** priority queue for integers with the normal `insert()` and `getMax()` operations, that is, whenever you call getMax(), it returns the largest integer in the queue.

Now suppose you have a sequence of insert(...) and getMax() operations in the following form:

```
// 0 or more statements of the form  System.out.print( getMax() + " ");
      insert( 0 );
// 0 or more statements of the form  System.out.print( getMax() + " ");
      insert( 1 );
// 0 or more statements of the form  System.out.print( getMax() + " ");
      insert( 2 );

            etc.

// 0 or more statements of the form  System.out.print( getMax() + " ");
      insert( 9 );
// 0 or more statements of the form  System.out.print( getMax() + " ");
```

That is, you insert the digits 0 ... 9 in order, and then before, in between, and after these inserts, you do some number of getMax() operations (possibly 0).

(A) 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.

(i) 1 3 5 7 6 4 9 8 2 0 POSSIBLE

(ii) 4 5 3 6 7 2 9 1 8 0 NOT POSSIBLE

(iii) 2 4 6 5 7 9 3 8 1 0 NOT POSSIBLE

(iv) 1 4 3 5 7 8 6 9 2 0 POSSIBLE

(B) Would your answers have been different if this question started "Suppose you have a **Stack** with the normal push(...) and pop() operations..."? Answer "yes" or "no" and explain briefly.

NO. SINCE ADDING IN ORDER 0 .... 9, NEW INSERT ALWAYS GOES TO FRONT, SO EXACTLY LIKE STACK!

Problem 5. For each function *f* from the following list of functions, determine the *simplest* function **g** that makes the equality $f(N) = O(g(N))$ true. Represent your answer as an equality (e.g. $f(N) = O(N^2)$). The first one is done for you as a model for the rest of your answers.

$a(N) = 2^N + N^2 - 5$ $\quad\quad$ $a(N) = O(2^N)$

$b(N) = \log(N^3) + \log(2N)$ $\quad\quad$ $b(N) = O(\log(N))$

$= 3*\log(N) + \log(2) + \log(N)$

$c(N) = 8N + 5*\text{sqrt}(N)$ $\quad\quad$ $c(N) = O(N)$

$= 8N + 5N^{\frac{1}{2}}$

$d(N) = 12N + N*\text{sqrt}(N) + N*\log(N).$ $\quad\quad$ $d(N) =$ [illegible]

$O(N^{1.5})$

$e(N) = \log^3(N) + \log(N) + 5$

$e(N) = O(\log^3(N))$

$f(N) = 50 + \frac{1}{N}$ $\quad\quad$ $F(N) = O(1)$

Problem 6. This is a question about Java and has two parts.

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

```
int a = 9;
double b = a / 2;
int c = 'C' - 'A';
String d = 4 + ("hi" + 7);
int e = (int) ( 7 / (double) 3);
a = ((10*a) % 6) % 5);

System.out.println("a = " + a);
System.out.println("b = " + b);
System.out.println("c = " + c);
System.out.println("d = " + d);
System.out.println("e = " + e);
```

INT DIVISION

67 - 65

STRING CONCAT.

2.3 ⇒ 2

(60 % 6)

0 % 5 = 0

0

4.0

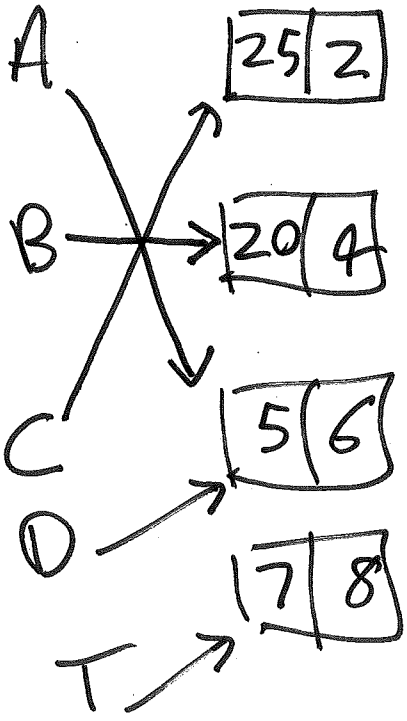
2

4hi7

2

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

```
int[] A = { 1, 2 };
int[] B = { 3, 4 };
int[] C = { 5, 6 };
int[] D = { 7, 8 };

int[] T = D;
D = C;
C = T;

A[0] = B[0] + C[0];

C = A;

B[0] = A[0] + C[0];

A = D;

C[0] = B[0] + A[0];

System.out.println(A[0]);
System.out.println(B[0]);
System.out.println(C[0]);
```



5

20

25

Problem 7. (MANDATORY)

(A) Complete the following method, which reverses a list of arbitrary size, under the following assumptions:

(a) 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;

(b) You must perform the absolute minimum number of swaps possible; your algorithm should work whether the array has even length or odd length.

```
// Reverse the integer array A

void reverse( int[] A ) {
        FOR(INT I=0; I<(A.LENGTH/2); ++I)
    ——→ IF( A[I] != A[A.LENGTH-1-I] )
            SWAP( A, I, (A.LENGTH-1-I) );
}

void swap(int[] A, int i, int j) {
     int temp = A[i];
     A[i] = A[j];
     A[j] = temp;
}
```

NICE TOUCH WHICH I DID NOT EXPECT!

(B) Suppose that we define the time complexity of your algorithm as the number of calls to `swap(...)` to reverse an array of size N. State PRECISELY how many swaps are performed by your algorithm as a function of N (you may separately give the complexity for odd N and even N).

$$\#\text{SWAPS} = \begin{cases} N/2 & \text{EVEN } N \\ \lfloor N/2 \rfloor & \text{ODD } N \end{cases}$$

OR JUST $\lfloor N/2 \rfloor$ OR $N/2$ USING INT DIVISION

(C) Give the worst-case time complexity of your algorithm in terms of O(...), where the expression .... inside the parentheses is as simple as possible.

$O(N)$

(D) Under the same assumptions as (B), give the O(.....) average-case complexity of your algorithm. Is this the same or different than your answer for (C)? Answer "yes" or "no" and explain in a short sentence.

$O(N)$ SAME

AVG. CASE WILL STILL INVOLVE A CONSTANT FRACTION OF WHOLE LIST.

IF DON'T USE IF STATEMENT,