Name: ___________________________

CS 112—Midterm Exam—Summer I, 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 them. Problem 1 is worth 10 points, and all other problems are worth 18 points. Please write in pen if possible. Circle answers when they occur in the midst of a bunch of work. If you need more room, use the back of the sheet and tell me this on the front sheet.

Problem One. (True/False – MANDATORY) Write True or False to the left of each statement. **Solution: False statements are in red.**

1. Python is an example of a "strongly-typed" language.
2. If a class Pair contains a method getFirst() and you see an expression Pair.getFirst(), then you know that getFirst() must be declared as static inside the class Pair.
3. The code in the box will print out a 6.

```
int a = 5;
System.out.println( a++ );
```

4. In the Boolean expression (A || B), if the expression A evaluates to false, then the expression B will not be evaluated.
5. If a linked list contains an sequence of integers in order, then you can use binary search to find whether a particular integer is in the list.
6. Insertion Sort is difficult to implement "in place" (i.e., without using an extra array).
7. The worst case time of Insertion Sort is the same (in terms of O(....)) as the best case time of Selection Sort.
8. When performing a *widening conversion* you must cast the value to the new type or else you will get an error.
9. The type String is a reference type.
10. KISS stands for "Kode it Soon, Silly!" (not graded!)

Problem Six. This problem is about Java and has two parts.

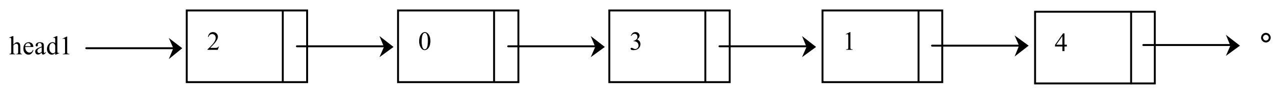
(A) What is the output of the following lines of Java code?

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

D = C;
A[1] = C[1] + B[1];
C = A;
B[1] = C[1] + A[1];
A = D;
C[1] = A[1] + B[1];

System.out.println(A[1]);        5

System.out.println(B[1]);        16

System.out.println(C[0]);        2
```

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

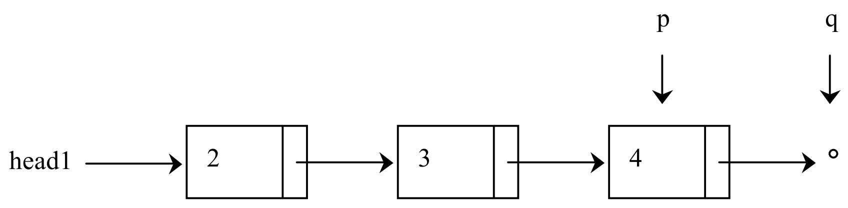
```
int a = 5;
double b = a / 2.0;
int c = a / 2;
double d = a / 2;
String e = c + "c";
a = a % 6;

System.out.println("a = " + a);        5

System.out.println("b = " + b);        2.5

System.out.println("c = " + c);        2

System.out.println("d = " + d);        2.0

System.out.println("e = " + e);        2c
```

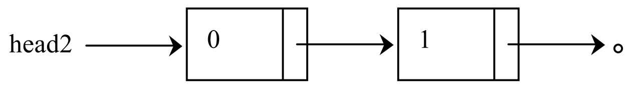
Problem Four. Consider this linked list:



Apply the following code to this linked list, and draw a picture of the data structure that results, including what the variables **head1**, **q**, **p**, and **head2** point to.

```
Node p = head1;
Node q = head1.next;
Node head2 = head1.next;

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





Problem Two. Consider the program shown below.

```
public class Mystery {

    public String a = "Hi there!";

    private int f(int b, int c) {
       int d = 4;
       for(int e = 0; e < 10; ++e) {
           if( c < 30 ) {
               double f = 2.3
           }
           else if( c < 20) {
               System.out.println( _____ );      // first blank
               double g = 9;
           }
        }
        System.out.println( _____ );             // second blank
    }

    private int h = 9;

    public double g(int i) {
        while( i < 20 ) {
           double j = 9.8;
        }
        System.out.println( _____ );             // third blank
    }

    private int k = 0;
}
```

(A) Which of the variables a -- k could be used in the first blank?

a, b, c, d, e, h, k

(B) Which of the variables a -- k could be used in the second blank?

a, b, c, d, h, k

(C) Which of the variables a -- k could be used in the third blank?

a, h, i, k

Problem Three. This is about **Mergesort** and has two parts.

(A) Perform Mergesort on the following array, according to the algorithm presented in lecture.

4 8 7 3 5 1 0 2

4 | 8 | 7 | 3 | 5 | 1 | 0 | 2

4 8 | 3 7 | 1 5 | 0 2

3 4 7 8 | 0 1 2 5

0 1 2 3 4 5 7 8

(B) What is the O(...N….) time complexity of just the merge operation of the algorithm?

O(N)

(C) What is the O(…N….) time complexity of the algorithm in the worst case?

O(N log(N))

(D) What is the O(…N….) time complexity of the algorithm in the average case?

O(N log(N))

(E) Why does mergesort require an auxiliary array, that is, why can it not be done with just a single array of size N?

A critical step in the algorithm is the merging of two sublists into a single list; in order to do this, you have to leave the original sublists in place, and copy the values to an auxiliary list as you merge them. This can not be done without auxiliary storage.

Problem Five. How many times does the following code print out “X”? Express your answer in terms of O(...N...).

[Hint: Write the O(...N...) estimate for each loop first, and then consider all loops together to get the final result. Partial credit will be given for the estimate for each loop!]

```
for( int a = 0; a < N; a = a + 2 ) {      O( N )
   for( int b = 0; b < a; ++b ) {         O( N )
      for( int c = 1; c < N;  c = c * 2 ) {O( log N )
          System.out.println(“X”);
      }
   }
}
for( int d = N; d >= 1; d = d / 2 ) { O( log N )
   for( int e = -N; e < N;  ++e ) {         O( N )
         System.out.println(“X”);
   }
}
```

Therefore you have to figure out $O(N * N * \log(N) + \log(N) * N)$

$$= O(N^2 \log(N))$$

Problem Seven. (MANDATORY) Suppose you have a queue data structure as follows:

```
public class StringQueue {
     public String dequeue(){…}
     public void enqueue(String s) {…}
     public int size(){…}
     public boolean sameEnds() { //not implemented!  }
}
```

You have implemented all but the last method, which returns **true** if the integers at the front and the rear of the queue are equal, **false** if the front and rear elements are not equal, and **false** if the queue has fewer than two elements, e.g.,

Use **equals(…)** to check for String equality, e.g., **s.equals(t).**

```
 rear                  front

  "hi"   "there"   "hi"      //   sameEnds() would return true

    "hi"     "there"       //   sameEnds() would return false

         "hi"            //   sameEnds() would return false ( < 2 elements )
```

Show how to implement sameEnds() using ONLY the other three methods; you may declare **String** variables but may not use an auxiliary array or create another queue. <u>NOTE: You must NOT destroy the data structure, i.e., it must be in the same exact configuration after calling the method sameEnds() as before.</u>

Solution:

```
public boolean sameEnds() {
    if(size() < 2)
       return false;
    String first = dequeue();
    enqueue(first);
    String last;
    int s = size();
    for(int i = 0; i < s-1; ++i) {
       last = dequeue();
       enqueue(last);
    }
    return (first.equals(last));
}
```