Name:

Instructions

1. Write your name at the top of the *first* page and your initials at the bottom of *every* page.
2. Do *not* staple the exam.
3. Return the exam with *all* the pages, arranged in *ascending* order.
4. This is a closed-book exam.
5. No electronic devices are permitted.
6. You may use the blank spaces for any scratch work.
7. Discussing the exam before the solutions have been posted is a violation of the Honor Code.
8. There are 11 problems on this exam and you have 75 minutes to answer them.
9. Problems 1 – 12 involve 19 multiple-choice questions, each worth 4 points. Each question must have *exactly one* response clearly marked in the circle provided — or else your answer will be considered incorrect.
10. Problems 13 (worth 16 points) and 14 (worth 8 points), must be answered clearly in the boxed space provided for those problems.

Problem 1. Consider the following method:

```
public static int mystery(int[][] a) {
    int x = 0;
    for (int i = 0; i < a.length; i++) {
        for (int j = 0; j < a[0].length; j++) {
            x += (i == j) ? a[i][j] * a[i][j] : 0;
        }
    }
    return x;
}
```

What does `mystery(a)` return, where `a = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}`?

- ◯ 45
- ◯ 83
- ● 107
- ◯ 126
- ◯ 66

Problem 2. Consider the following recursive method:

```
public static int mystery(int a, int b) {
    return (b == 0) ? a : mystery(b, a % b);
}
```

a. What does `mystery(27, 72)` return?

- ◯ 27
- ● 9
- ◯ 45
- ◯ 1
- ◯ 72

b. What does `mystery(48, 15)` return?

- ◯ 48
- ◯ 15
- ◯ 1
- ◯ 33
- ● 3

c. What does `mystery()` compute and return in general?

- ◯ b
- ● Largest number that divides both a and b
- ◯ Smallest number that divides both a and b
- ◯ $|a - b|$
- ◯ a

Problem 3. Consider the following recursive methods::

```
public static int f(Node x) {
    return (x == null) ? 0 : 1 + f(x.next);
}

public static int g(Node x) {
    return (x == null) ? 0 : g.item + g(x.next);
}
```

a. What does `f(a)` return, where `a` is a linked list containing the items 1, 1, 2, 3, 5, 8, and 13 and in that order?

- (●) `7`
- ( ) `33`
- ( ) `1`
- ( ) `13`
- ( ) `0`

b. What does `g(a)` return, where `a` is a linked list containing the items 1, 1, 2, 3, 5, 8, and 13 and in that order?

- ( ) `7`
- ( ) `1`
- ( ) `0`
- (●) `33`
- ( ) `13`

Problem 4. Consider the following program `Mystery.java`:

```
public class Mystery {
    public static void main(String[] args) {
        String x = StdIn.readString();
        String y = StdIn.readString();
        StdOut.print(x + y);
        StdOut.print(" ");
        StdOut.print(y + x);
        StdOut.println();
    }
}
```

Next, suppose that the file `input.txt` contains the two strings `AB` and `BA` separated by a space. What does the following command output?

```
$ java Mystery < input.txt | java Mystery | java Mystery
```

- ( ) `AB BA`
- ( ) `ABBA BAAB`
- (●) `ABBABAABBAABABBA BAABABBAABBABAAB`
- ( ) `ABBABAABBAABABBABAABABBAABBABAAB BAABABBAABBABAABABBABAABBAABABBA`
- ( ) `ABBABAAB BAABABBA`

Problem 5. Consider the following table, which gives the running time $T(N)$ in seconds for a program for various values of the input size N:

N	$T(N)$
1000	3
2000	12
4000	48
8000	192

What is the order of growth of $T(N)$?

- (●) Quadratic
- ( ) Linear
- ( ) Exponential
- ( ) Linearithmic
- ( ) Cubic

Problem 6. What is the order of growth ($T(N)$) of the following code fragment?

```
int sum = 0;
for (int i = 0; i < N; i++) {
    for (int j = 0; j < 100; j++) {
        for (int k = 0; k < 1000; k++) {
            sum++;
        }
    }
}
```

- ( ) Cubic
- (●) Linear
- ( ) Quadratic
- ( ) Exponential
- ( ) Linearithmic

Problem 7. Consider a data type `T` with two instance variables: `int x` and `double y`. Ignoring array and object overheads, what is the memory footprint (in bytes) of the array `a[]` created and initialized as follows?

```
T[] a = new T[100];
for (int i = 0; i < 100; i++) {
    T[i] = new T();
}
```

- ◯ 100
- ● 1200
- ◯ 12
- ◯ 400
- ◯ 800

Problem 8. Consider the following methods:

```
Iterator<Character> f(String s) {
    Queue<Character> Q = new Queue<Character>();
    for (int i = 0; i < s.length(); i++) { Q.enqueue(s.charAt(i)); }
    return Q.iterator();
}

Iterator<Character> g(String s) {
    Stack<Character> S = new Stack<Character>();
    for (int i = 0; i < s.length(); i++) { S.push(s.charAt(i)); }
    return S.iterator();
}
```

a. What is the value returned by the method call `f("alice").next()`?

- ◯ 'e'
- ◯ 'l'
- ◯ 'c'
- ● 'a'
- ◯ 'i'

b. What is the value returned by the method call `g("alice").next()`?

- ◯ 'i'
- ◯ 'l'
- ◯ 'c'
- ◯ 'a'
- ● 'e'

Problem 9. Suppose we use the `QuickUnionUF` data structure to solve the dynamic connectivity problem with 10 sites and input pairs $(8,1)$, $(7,6)$, $(9,2)$, $(7,8)$, $(4,6)$, $(6,0)$, and $(4,1)$, arriving in that order; the code for the `union()` method in `QuickUnionUF` is shown below.

```
public void union(int p, int q) {
    int rootP = find(p);
    int rootQ = find(q);
    if (rootP == rootQ) { return; }
    parent[rootP] = rootQ;
    count--;
}
```

a. What are the values in the `parent` array after all the pairs are processed?

- ○ `parent = {0, 0, 2, 0, 0, 2, 6, 0, 8, 0}`
- ● `parent = {0, 0, 2, 3, 1, 5, 1, 6, 1, 2}`
- ○ `parent = {0, 0, 0, 3, 2, 2, 0, 0, 0, 9}`
- ○ `parent = {2, 0, 2, 0, 0, 0, 6, 0, 8, 0}`
- ○ `parent = {0, 1, 0, 2, 0, 0, 0, 2, 8, 0}`

b. What is the size of the largest component?

- ○ 2
- ○ 5
- ○ 3
- ○ 4
- ● 6

c. What is the identifier of the largest component?

- ○ 2
- ○ 3
- ○ 1
- ● 0
- ○ 4

Problem 10. Consider sorting an array `a[]` containing the following strings, using selection sort (shown below):

```
C   G   H   Y   V   T   S   M   Z   N
```

```
public static void sort(Comparable[] a) {
    int N = a.length;
    for (int i = 0; i < N; i++) {
        int min = i;
        for (int j = i + 1; j < N; j++) {
            if (less(a[j], a[min])) {
                min = j;
            }
        }
        exch(a, i, min);
    }
}
```

What is the value that `Y` is exchanged with?

- ◯ `H`
- ◯ `S`
- ● `M`
- ◯ `Z`
- ◯ `N`

Problem 11. Consider sorting an array `a[]` containing the following strings, using insertion sort (shown below):

```
D  J  M  R  S  T  Y  Z  O  F
```

```
public static void sort(Comparable[] a) {
    int N = a.length;
    for (int i = 1; i < N; i++) {
        for (int j = i; j > 0 && less(a[j], a[j - 1]); j--) {
            exch(a, j, j - 1);
        }
    }
}
```

Where is the item `O` sorted (ie, what is its index) relative to the items before?

- ◯ 1
- ◯ 4
- ● 3
- ◯ 0
- ◯ 2

Problem 12. Consider sorting an array `a[]` containing the following strings, using quick sort (shown below):

```
V  U  Z  L  S  Y  R  E  I  J
```

```
public static void sort(Comparable[] a) {
    sort(a, 0, a.length - 1);
}

private static void sort(Comparable[] a, int lo, int hi) {
    if (hi <= lo) return;
    int j = partition(a, lo, hi);
    sort(a, lo, j - 1);
    sort(a, j + 1, hi);
}

private static int partition(Comparable[] a, int lo, int hi) {
    int i = lo;
```

```
    int j = hi + 1;
    Comparable v = a[lo];
    while (true) {
        while (less(a[++i], v)) { if (i == hi) { break; } }
        while (less(v, a[--j])) { if (j == lo) { break; } }
        if (i >= j) { break; }
        exch(a, i, j);
    }
    exch(a, lo, j);
    return j;
}
```

a. What is the state of the array `a` after the first call to `partition()`?

- (●) E U J L S I R V Y Z
- ( ) I E J R U S L V Z Y
- ( ) I S U R J L E V Y Z
- ( ) E J I S R U L V Y Z
- ( ) R L U J E I S V Z Y

b. What is pivot element in the next call to `partition()`?

- ( ) Z
- (●) E
- ( ) R
- ( ) I
- ( ) Y

Problem 13. Implement a comparable and iterable data type `Genome` that represents a genome sequence (a string of letters A, T, G, or C denoting nucleotides), and supports the following API:

	method	**description**
	`Genome(String s)`	construct a genome from s
a. (2 points)	`double gcContent()`	$\frac{G+C}{A+T+G+C} \times 100$; for example, GC content of the genome sequence "ACTGCG" is 67%
b. (2 points)	`int compareTo(Genome that)`	a comparison of *this* and *that* genome by their lengths
c. (2 points)	`static GCContentOrder`	for comparing genome sequences by their GC content
d-f (6 points)	`Iterator iterator()`	for iterating over the genome in *reverse* order

If `s` is a `String` object, you may use `s.charAt(i)` to obtain the ith character in the string.

```
import java.util.Comparator;
import java.util.Iterator;

public class Genome implements Comparable<Genome>, Iterable<Character> {
    private final String s; // the genome sequence

    public Genome(String s) {
        this.s = s;
```

```
    }

    public double gcContent() {
        int gc = 0;
        for (int i = 0; i < s.length(); i++) {
            if (s.charAt(i) == 'G' || s.charAt(i) == 'C') { gc++; }
        }
        return 100.0 * gc / s.length();
    }

    public int compareTo(Genome that) {
        return this.s.length() - that.s.length();
    }

    public static class GCContentOrder implements Comparator<Genome> {
        public int compare(Genome g1, Genome g2) {
            double x = g1.gcContent();
            double y = g2.gcContent();
            if (x < y)       { return -1; }
            else if (x == y) { return 0; }
            else             { return 1; }
        }
    }

    public Iterator<Character> iterator() { return new ReverseIterator(); }

    private class ReverseIterator implements Iterator {
        private int i; // index of current letter

        public ReverseIterator() { this.i = s.length() - 1; }

        public boolean hasNext() { return i >= 0; }

        public Character next() { return s.charAt(i); }

        public void remove() {}
    }
}
```

g. (2 points) Suppose `sequences` is an array of `Genome` objects. Write down a statement that uses `Arrays.sort()` to sort `sequences` by length.

```
Arrays.sort(sequences);
```

h. (2 points) Write down a statement that uses `Arrays.sort()` to sort `sequences` by GC content.

```
Arrays.sort(sequences, new Genome.GCContentOrder());
```

Problem 14. a. (6 points) Given an array `a` containing N integers, provide a crisp and concise English description of an algorithm for finding the *closest* pair of integers. For example, `(3, 4)` is the closest pair in the array `a = {4, 9, 3, -1, 6}`.

> Sort the array `a` using quick sort and then scan through the array to find successive integers a and b such that $b - a$ is minimum. The pair (a, b) is the closest pair.

b. (2 points) What is the order of growth of the worst case running time of your algorithm?

$N \log N$