

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 13 problems on this exam and you have 180 minutes to answer them.
9. Problems 1 – 11 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 marked incorrect.
10. Problems 12 (worth 16 points) and 13 (worth 8 points), must be answered clearly in the boxed space provided for those problems.

Problem 1. Consider the following recursive method:

```
public static int mystery(int a, int b) {  
    if (b == 0) {  
        return 1;  
    }  
    return a * mystery(a, b - 1);  
}
```

a. What does `mystery(3, 5)` return?

- ☐ 125
- ☐ 5
- ☐ 15
- ☐ 3
- ☐ 243

Initials:

b. What does `mystery(a, b)` return in general about a and b ?

- ☐ a
- ☐ a^b
- ☐ b
- ☐ b^a
- ☐ ab

Problem 2. 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	24
4000	192
8000	1536

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

- ☐ Linearithmic
- ☐ Linear
- ☐ Quadratic
- ☐ Cubic
- ☐ Logarithmic

Problem 3. What does the following code fragment write when n is 113?

```
Stack<Integer> s = new Stack<Integer>();
while (n > 0) {
    s.push(n % 2);
    n = n / 2;
}
while (!s.isEmpty()) {
    StdOut.print(s.pop());
}
```

- ☐ 1110101
- ☐ 1101001
- ☐ 1010001
- ☐ 1110011
- ☐ 1110001

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

```
public void union(int p, int q) {
    int pID = id[p];
    int qID = id[q];
    if (pID == qID) {
        return;
    }
    for (int i = 0; i < id.length; i++) {
        if (id[i] == pID) {
            id[i] = qID;
        }
    }
    count--;
}
```

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

- ☐ `id = {0, 0, 2, 3, 0, 5, 0, 0, 0, 2}`
- ☐ `id = {0, 1, 0, 2, 0, 0, 0, 2, 8, 0}`
- ☐ `id = {2, 0, 2, 0, 0, 0, 6, 0, 8, 0}`
- ☐ `id = {0, 0, 2, 3, 1, 5, 1, 6, 1, 2}`
- ☐ `id = {0, 0, 2, 0, 0, 2, 6, 0, 8, 0}`

b. What is the identifier of the largest component?

- ☐ 4
- ☐ 2
- ☐ 1
- ☐ 3
- ☐ 0

Problem 5. Consider sorting an array `a[]` containing the following strings, using quick sort (shown below):

P T H C N R Q X Z V

```
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 final (destination) index of the pivot element P after the first call to `partition()`?

- ☐ 3
- ☐ 2
- ☐ 1
- ☐ 4
- ☐ 5

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

- ☐ T
- ☐ P
- ☐ H
- ☐ N
- ☐ C

Problem 6. Insert the following keys in that order into a minimum-oriented heap-ordered binary tree:

C Y V J P G Q F H

If we perform a `delMin()` operation on the tree, what is the key that will replace the current minimum (the key c) before it is sunk down?

- ☐ P
- ☐ V
- ☐ J
- ☐ Y
- ☐ Q

Problem 7. Consider inserting the following keys (assume values to be non null and arbitrary) into a binary search tree (BST) symbol table `st`, an object of type `BST`.

I K C W U R F L E T N A Q Z

a. What is the height of the BST (assume root to be at height 0)?

- ☐ 7
- ☐ 5
- ☐ 8
- ☐ 4
- ☐ 6

b. What is the order in which the keys are visited if we traverse the BST in pre-order?

- ☐ I C A F E K W U R T L Q Z N
- ☐ I C A F E K W U R L Z T N Q
- ☐ I C A F E K W U R L N Q T Z
- ☐ I C K A F W E U Z R L T N Q
- ☐ A C E F I K L N Q R T U W Z

Problem 8. Consider inserting the following keys into an initially empty 2-3 search tree:

J I B Y F W V U A P L

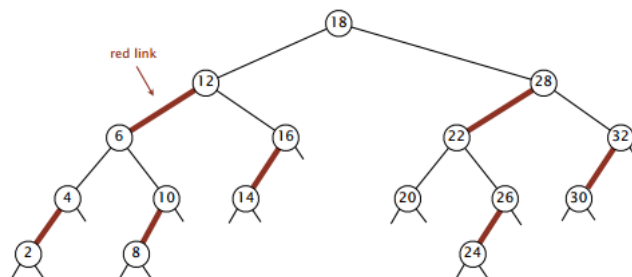
a. How many 3-nodes does the tree contain?

- ☐ 4
- ☐ 2
- ☐ 0
- ☐ 1
- ☐ 3

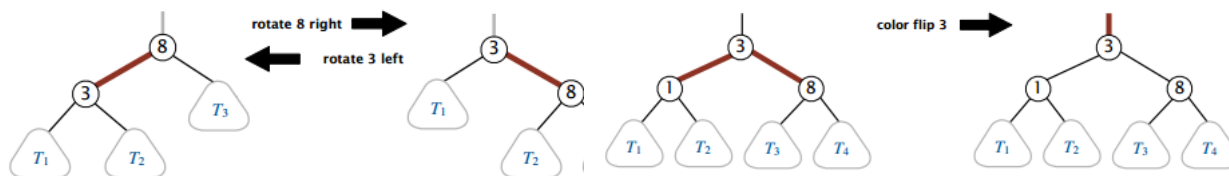
b. Which of the following keys is part of the root node?

- ☐ U
- ☐ A
- ☐ F
- ☐ L
- ☐ P

Problem 9. Suppose you insert the key 11 into the following left-leaning red-black BST:



Allowed operations (rotations and color flip):



a. What is the *first* operation that results?

- ☐ Rotate 12 left
- ☐ Rotate 6 left
- ☐ Color flip 10
- ☐ Rotate 6 right
- ☐ Rotate 12 right

b. What is the *third* operation that results?

- ☐ Rotate 12 left
- ☐ Rotate 12 right
- ☐ Color flip 10
- ☐ Rotate 6 right
- ☐ Rotate 6 left

Problem 10. Consider inserting the following keys (assume values to be non null and arbitrary) into an initially empty hash table of $M = 5$ lists, using separate chaining. Use the hash function $h(k) = k \bmod M$ to transform the k th letter of the alphabet into a table index, where $1 \leq k \leq 26$.

C J F Y K H P S Q R G U

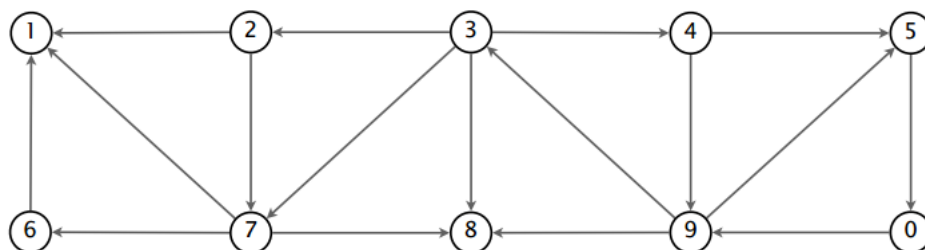
a. What is the index of the longest chain?

- ☐ 3
- ☐ 4
- ☐ 1
- ☐ 2
- ☐ 0

b. Which of the following keys is *not* in the longest chain?

- ☐ K
- ☐ Q
- ☐ U
- ☐ P
- ☐ F

Problem 11. Assume that the adjacency lists for the following digraph are in sorted order: for example, when iterating over the edges pointing from 3, process the edge $3 \rightarrow 2$ before either $3 \rightarrow 7$ or $3 \rightarrow 8$.



a. Do a breadth-first search with 0 as the source vertex, and list the order in which vertices are processed by the algorithm?

- ☐ 0 9 3 6 1 4 5 2 7 8
- ☐ 0 9 3 7 4 1 2 6 8 5
- ☐ 0 9 3 5 8 2 4 7 1 6
- ☐ 0 9 3 2 7 8 6 5 1 4
- ☐ 0 9 3 2 7 8 1 6 5 4

b. Do a depth-first search with 0 as the source vertex, and list all vertices in pre-order.

- ☐ 0 9 3 2 4 6 1 5 8 7
- ☐ 0 9 3 2 1 7 6 8 4 5
- ☐ 0 9 3 6 1 5 4 7 8 2
- ☐ 0 9 3 8 2 6 4 5 1 7
- ☐ 0 9 3 7 8 4 1 2 6 5

Problem 12. (16 points) Implement a comparable data type `Country` that represents a country, and supports the following API:

	Method/Class	Description
a. (2 points)	<code>Country(String name, String code, long population)</code>	construct a country given its name, code, and population
b. (2 points)	<code>boolean equals(Country that)</code>	is this country's code the same as that country's code?
c. (2 points)	<code>String toString()</code>	a string representation in the format "name, code, population"
d. (2 points)	<code>int compareTo(Country that)</code>	comparison of this and that country by name
e. (2 points)	<code>static class ReverseCodeOrder</code>	a comparator for comparing two countries in reverse order of their codes
f. (2 points)	<code>static class PopulationOrder</code>	a comparator for comparing two countries by population

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

```
public class Country implements Comparable<Country> {  
    private final String name, code;  
    private final long population;
```

```
    public Country(String name, String code, long population) {
```

```
}
```

```
    public boolean equals(Country that) {
```

```
}
```

```
    public String toString() {
```

```
}
```



```
public int compareTo(Country that) {
```

```
}
```

```
public static class ReverseCodeOrder implements Comparator<Country> {  
    public int compare(Country x, Country y) {
```

```
    }
```

```
}
```

```
public static class PopulationOrder implements Comparator<Country> {  
    public int compare(Country x, Country y) {
```

```
    }
```

```
}
```

```
}
```

g. (2 points) Suppose `countries` is an array of `Country` objects. Write down a statement that uses `Arrays.sort()` to sort `countries` by name.

h. (2 points) Write down a statement that uses `Arrays.sort()` to sort `countries` by population.

Problem 13. a. (6 points) Given an array a containing N integers, provide a crisp and concise English description of an algorithm for finding the *farthest* pair of integers. For example, $(-1, 9)$ is the farthest pair in the array $a = \{4, 9, 3, -1, 6\}$.

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