

Name: _____ Section: _____ CM: _____

KEY

CSSE 220—Object-Oriented Software Development

Final Exam – Part 1, November 16, 2016

This exam consists of two parts. Part 1 is to be solved on these pages. You may use the back of a page if you need more room. Please indicate on the front if you do so. Part 2 is to be solved on your computer. You will need network access to download template code and upload your solution for part 2. Please disable IM, email, Lync, and other such communication programs before beginning the exam.

Resources for Part 1: One 8 1/2" by 11" double-sided sheet of notes, closed book, closed computer, closed electronic devices.

Resources for Part 2: Open book, open notes, and open computer. Limited network access. You may use the network only to access your own files and Moodle sites, the course web pages, the textbook's site, and Oracle's Java website (including API documentation).

Any communication with anyone other than an exam proctor during the exam may result in a failing grade for the course.

Part 1 is included in this document. You should read over all of the questions before beginning work, but...

You must turn in part 1 before accessing the resources for part 2.

Problem	Poss. Pts.	Earned
1	6	_____
2	8	_____
3	3	_____
4	6	_____
5	12	_____
Paper Part Subtotal	35	_____
Computer Part Subtotal	65	_____
Total	100	_____

Part 1—Paper Part

1. (6 points) Consider the following initial array configuration. In each question, assume we want to sort the array in **alphabetic** order (that is, **from A – Z first, then from a – z**)., so Z comes before a.

f	A	m	a	x	B	t	h	Z	g
---	---	---	---	---	---	---	---	---	---

- a. (3 points) Suppose the **insertion** sort algorithm from class is applied to the **initial array above**. Show the state of the array immediately following **each of the first three iterations** of the outer loop. **Clearly mark** the sorted portion and the unsorted portion of the array. Note: the sorted part of the array initially contains **1 element**.

1st iteration:

A	f		m	a	x	B	t	h	Z	g
Sorted										

2nd iteration:

A	f	m		a	x	B	t	h	Z	g
Sorted										

3rd iteration:

A	a	f	m		x	B	t	h	Z	g
Sorted										

- b. (1 point) Suppose the **merge** sort algorithm from class is applied to the **initial array above**. Show the state of the two sub-arrays immediately before the final merge.

A	a	f	m	x		B	Z	g	h	t
---	---	---	---	---	--	---	---	---	---	---

- c. (2 points) Suppose the **selection** sort algorithm from class is applied to the **initial array above**. In the boxes below, show the state of the array immediately following **each of the first two iterations** of the outer loop. **Clearly mark** the sorted portion and the unsorted portion of the array. Note: the sorted part of the array initially contains **NO elements**.

1st iteration:

A	f	m	a	x	B	t	h	Z	g
Sorted									

2nd iteration:

A	B		m	a	x	f	t	h	Z	g
Sorted										

if sort from back, OK

g		x		
Z		t		x

2. (8 points) Answer the questions below on the sorting, searching and data structures that we discussed in class.

a. For each of the algorithms below, write the Big-O runtime in the **best** case.

Adding an element to an ArrayList $O(1)$

Selection Sort $O(n^2)$

Merge Sort $O(n \log n)$

Search for an element in LinkedList (no tail pointer) $O(1)$

b. For each of the algorithms below, write the Big-O runtime in the **worst** case.

Binary Search $O(\log n)$

Insertion Sort $O(n^2)$

Remove an element from an ArrayList $O(n)$

Search for an element in a LinkedList (no tail pointer) $O(n)$

3. (3 points) Predict the output for the code below.

```

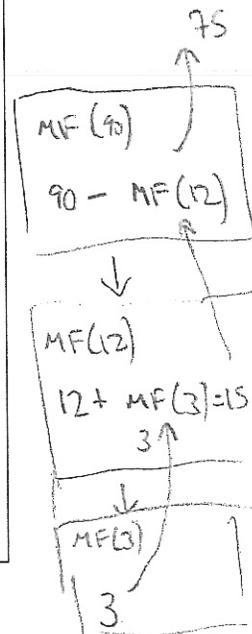
public class MysteryFunction {
    public int mysteryFunc(int input) {
        if (input <= 10) {
            return input;
        }
        int result = input;
        if(input % 10 == 0)
            result -= mysteryFunc(input/7); //integer division!!
        else
            result += mysteryFunc(input-9);

        return result;
    }

    public static void main(String[] args) {
        MysteryFunction m = new MysteryFunction();
        System.out.println("The result is: " + m.mysteryFunc(90));
    }
}

```

Output: The result is 75



4. (6 points) Give the Big-O runtime for each of the code snippets below. Answers are worth 2 points each.

a.

```
public void firstAlg(int n) {
    for(int i = 0; i < 900000000; i++) {
        n += i;
    }
}
```

Answer: $O(1)$

b.

```
public void secondAlg(int n) {
    int x = 0;
    for(k=0; k<n; k++) {
        while(x < n) {
            x = x * 2;
        }
    }
}
```

Answer: $O(n/\log n)$

c.

Assume $\text{data.size()} \geq n^2$

```
public void thirdAlg(ArrayList<String> data) {
    long n = data.size();
    for (long i = 0; i < n; i++) {
        for (long j = 0; j < (n*n); j++) {
            System.out.print(data.get(j) + " ");
        }
        System.out.println(data.get(i));
    }
}
```

Answer: $O(n^3)$

$n \binom{n^2}{n}$

5. (12 points) Use the code below to answer the associated questions.

```
public interface A {
    public void doubleVal(int input);
}

public abstract class B implements A {
    @Override
    public void doubleVal(int input) {
        System.out.println(input*2);
        this.repeat(input);
    }

    public abstract void repeat(int input);

    public void getSquare(int input) {
        System.out.println(input*input);
    }
}

public class C extends B {
    @Override
    public void repeat(int input){
        System.out.println(input+ "OKaay!");
    }
}
```

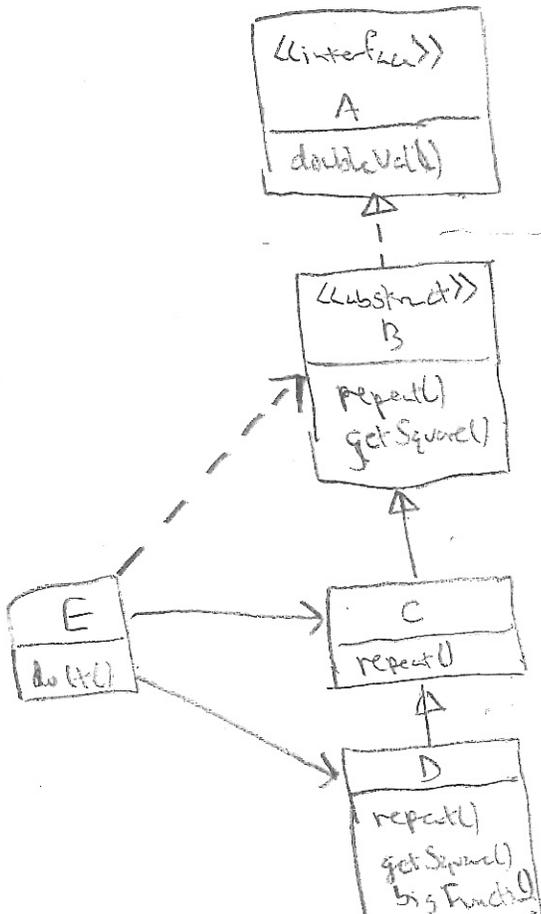
```
public class D extends C {
    @Override
    public void repeat(int input) {
        System.out.println("Yayah!!!");
    }

    @Override
    public void getSquare(int input) {
        System.out.println("WHAT!!"+input);
    }

    public void bigFunction(B input) {
        input.repeat(123);
    }
}

public class E {
    private C cVal = new C();
    private D dVal = new D();
    public void dolt(B input) {
        cVal.getSquare(4);
        dVal.bigFunction(cVal);
    }
}
```

a. (4 Points) Draw a UML class diagram that shows the relationships among the above classes.



1 classes
 2 arrow types
 1 arrow locations

b. (8 points) Using the declarations from the previous classes: Consider each of the code snippets independently (that is, errors in one do not affect the others). For each, write the output. If the code snippet results in an error, indicate the type of error (either Compile-time or Runtime error).

- | | | |
|------|---|-----------------------|
| i | <pre>C c = new C();
c.doubleVal(14);</pre> | <u>28 / 14 Okay!</u> |
| ii | <pre>A a = new B();
a.getSquare(2);</pre> | <u>Compile error</u> |
| iii | <pre>A a = new D();
a.doubleVal(9);</pre> | <u>18 / Yayah!!!</u> |
| iv | <pre>B b = new D();
b.bigFunction(new C());</pre> | <u>Compile error</u> |
| v | <pre>E e = new E();
e.dolt(new C());</pre> | <u>16 / 123 Okay!</u> |
| vi | <pre>B b = new C();
b.repeat(4);</pre> | <u>4 Okay!</u> |
| vii | <pre>C c = new C();
((D) c).bigFunction(new C());</pre> | <u>runtime error.</u> |
| viii | <pre>B b = new D();
b.getSquare(4);</pre> | <u>WHAT!! 4</u> |

