

Part 1—Paper Part

1. (8 points) Consider the following initial array configuration. In each question, assume we want to sort the array in **ascending** order (that is, from smallest to largest).

4	5	1	7	3	2	8	6
---	---	---	---	---	---	---	---

- 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 the first **three** executions of the outer loop.

1	4	5	7	3	2	8	6
---	---	---	---	---	---	---	---

after 1: 4 5 1 7 3 2 8 6

after 2: 1 4 5 7 3 2 8 6

after 3: 1 4 5 7 3 2 8 6

- b. (3 points) Suppose the **selection** sort algorithm from class is applied to the **initial array above**. Show the state of the array immediately following the first **three** executions of the outer loop.

1	2	3	7	4	5	8	6
---	---	---	---	---	---	---	---

after 1: 1 5 4 7 3 2 8 6

after 2: 1 2 4 7 3 5 8 6

after 3: 1 2 3 7 4 5 8 6

- c. (2 points) 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.

1	4	5	7
---	---	---	---

2	3	6	8
---	---	---	---

2. (8 points) For each of the scenarios below, indicate which of the *data structures* we studied—**array list, linked list, stack, queue, hash set, tree set, hash map, or tree map**—would be best. Justify your answer.

- a. Modeling my personal todo list, where every item has a priority and I always want it to remain sorted (also, all items are unique)

Data Structure: tree set

Justification:

ensures items are unique and sorted

- b. Maintaining a list of files to process, where newly added files need to wait for all the files currently in the list to finish processing before they can be started

Data Structure: queue

Justification:

FIFO is what we want

- c. Keeping track of student birthdays, where I want to easily be able to look up a birthday given a student name

Data Structure: hash map

Justification:

$O(1)$ lookup time

- d. An ordered list of elements that I rarely add or remove elements to, but often need to access by index (e.g. the 5th list element, the 234th element, etc.)

Data Structure: array list

Justification:

$O(1)$ index lookup time

3. (6 points) Predict the output of each of the following code blocks:

a.

```
Stack<Integer> s = new Stack<Integer>();
s.push(1);
s.push(2);
System.out.print(s.pop()); 2
System.out.print(s.pop()); 1
s.push(3);
System.out.print(s.pop()); 3
s.push(4);
s.push(5);
System.out.print(s.pop()); 5
System.out.print(s.pop()); 4
```

Answer: 2 1 3 5 4

b.

```
Queue<Integer> s = new Queue<Integer>();
// actually Java's queue uses slightly different
// methods but these are the ones from class
// just in case you forgotten enqueue = offer
// dequeue = poll
s.enqueue(10);
s.enqueue(9);
System.out.print(s.dequeue()); 10
System.out.print(s.dequeue()); 9
s.enqueue(8);
System.out.print(s.dequeue()); 8
s.enqueue(7);
s.enqueue(6);
System.out.print(s.dequeue()); 7
System.out.print(s.dequeue()); 6
```

Answer: 10 9 8 7 6

c.

```
HashMap<Character,String> map =
    new HashMap<Character, String>();

map.put('x', "Ninja");
map.put('y', "Pirate");
map.put('z', "Robot");

System.out.print(map.get('z'));
System.out.print(map.get('x'));
```

Answer: Robot Ninja

4. (10 points) Write the Big-O of each of the following code blocks:

a.

```
public void fun1(int n) {  
    if(n % 7 == 0)  
        return;  
    for(int i = 0; i < n; i++) {  
        for(int j = 0; j < n; j++) {  
            System.out.println("fun1");  
        }  
    }  
}
```

Answer:

$O(n^2)$

b.

```
public void fun2(int n) {  
    for(int i = 0; i < n; i++) {  
        for(int j = 0; j < n-i; j++) {  
            System.out.println("fun2");  
        }  
    }  
}
```

Answer:

$O(n^2)$

c.

```
// n is the number of elements in the list  
public int fun3(LinkedList<Integer> list) {  
    int n = list.size();  
    //returns the middle element  
    return list.get(n/2);  
}
```

Answer:

$O(n)$

d.

```
// n is the number of elements in the array  
public int fun4(int[] list) {  
    for(int i = 0; i < list.length; i++) {  
        doABinarySearch(list, 44);  
    }  
}
```

Answer:

$O(n \log n)$

e.

```
public void fun5(int n) {
    int data = n;
    while(data > 3) {
        data = data/2;
        System.out.println("Foo!");
    }
}
```

Answer: $O(\log n)$

5. (9 points) What does the following code print?

a.

```
fun6("abc");

// elsewhere...
public void fun6(String input) {
    if(input.isEmpty())
        return;
    char c = input.charAt(0);
    System.out.print(c);
    fun6(input.substring(1));
    System.out.print(c);
}
```

Answer:

a b c c b a

b.

```
int[] data = {1,2,6,7,8,9};
int[] result = fun7(data, 0);
for(int i = 0; i < result.length; i++) {
    System.out.print(result[i] + ", ");
}

//elsewhere...
public int[] fun7(int[] data, int i) {
    int a = data[i];
    if(a == 6) return data;
    if(i > data.length/2) return data;

    data[i] = data[data.length - 1 - i];
    data[data.length - 1 - i] = a;
    return fun7(data, i+1);
}
```

Answer:

9, 8, 6, 7, 2, 1,

my work ↴

9, 2, 6, 7, 8, 1
9, 8, 6, 7, 2, 1

c.

```
System.out.println(fun8(28));
```

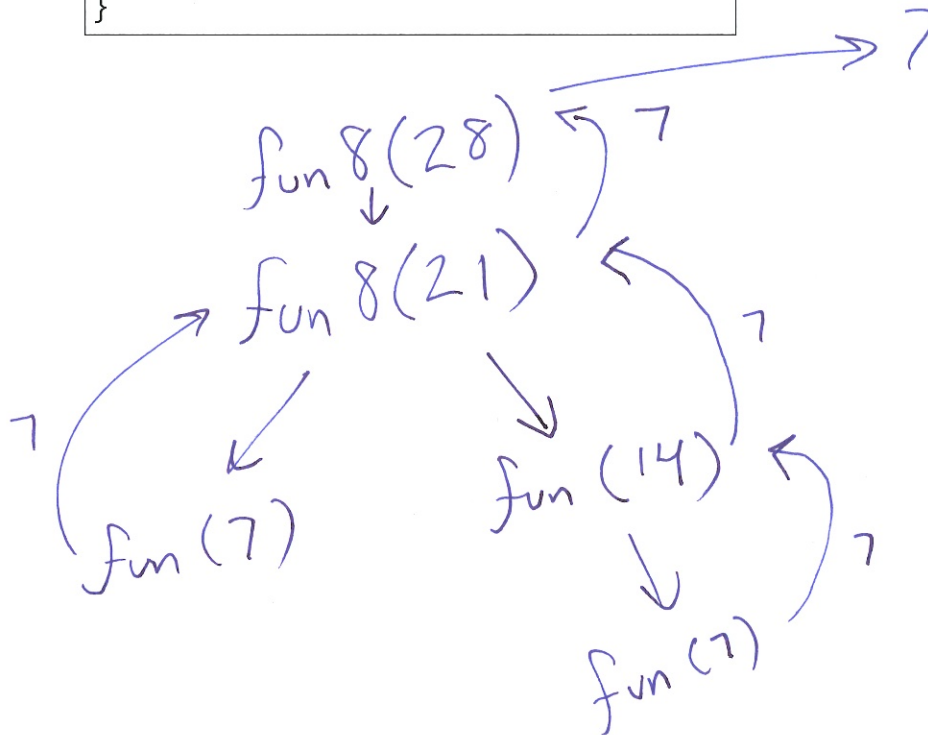
```
//elsewhere...
```

```
public int fun8(int input) {  
    int min = input;  
    if(input > 9 && input % 3 == 0) {  
        min = Math.min(min, fun8(input / 3));  
    }  
  
    if(input > 10) {  
        min = Math.min(min, fun8(input - 7));  
    }  
    return min;  
}
```

Answer:

7

My work:



6. (10 points) Consider the following related declarations:

```
class A {  
  
    public void number()  
    {  
        System.out.println(1);  
    }  
  
    public void letter()  
    {  
        System.out.println("a");  
    }  
}  
  
class B extends A {  
    public void letter()  
    {  
        System.out.println("b");  
    }  
  
    public void letter2()  
    {  
        System.out.println("B");  
    }  
}
```

```
class C extends A { }  
  
abstract class D extends C {  
    abstract public void l3();  
}  
  
class E extends D {  
    public void l3()  
    {  
        System.out.println("E");  
    }  
}
```


Suppose we declare and initialize these variables:

```
A a = new A();
A a2 = new B();
B b = new B();
C c = new C();
```

For each line of code below, **circle** what would be output. If the line would work but not output anything, circle nothing. If a line would give an error, then circle the type of error. Consider each line separately. That is, if a line would give an error, then assume that line doesn't affect any others.

Code	Output Choices (circle one in each line)
a2.letter();	a <input checked="" type="radio"/> b B 1 E <i>nothing</i> <i>runtime error</i> <i>compile error</i>
a2.letter2();	a b B 1 E <i>nothing</i> <i>runtime error</i> <input checked="" type="radio"/> <i>compile error</i>
C c1 = new B();	a b B 1 E <i>nothing</i> <i>runtime error</i> <input checked="" type="radio"/> <i>compile error</i>
C c2 = new A();	a b B 1 E <i>nothing</i> <i>runtime error</i> <input checked="" type="radio"/> <i>compile error</i>
C c3 = (C) new A();	a b B 1 E <i>nothing</i> <input checked="" type="radio"/> <i>runtime error</i> <i>compile error</i>
b.number();	a b B <input checked="" type="radio"/> 1 E <i>nothing</i> <i>runtime error</i> <i>compile error</i>
a = b;	a b B 1 E <input checked="" type="radio"/> <i>nothing</i> <i>runtime error</i> <i>compile error</i>
c.letter2();	a b B 1 E <i>nothing</i> <i>runtime error</i> <input checked="" type="radio"/> <i>compile error</i>
D d = new D(); d.l3();	a b B 1 E <i>nothing</i> <i>runtime error</i> <input checked="" type="radio"/> <i>compile error</i>
D d2 = new E(); d2.l3();	a b B 1 <input checked="" type="radio"/> E <i>nothing</i> <i>runtime error</i> <i>compile error</i>