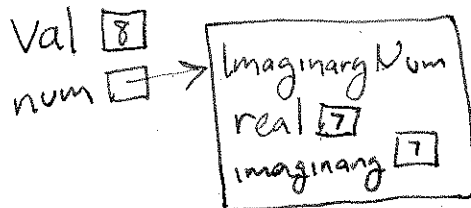


1. (9 points) Below are several code snippets that use the ImaginaryNum class. For each snippet, first draw a box-and-pointer diagram showing the result of executing it. Then give the output of the print statement at the end of the snippet.

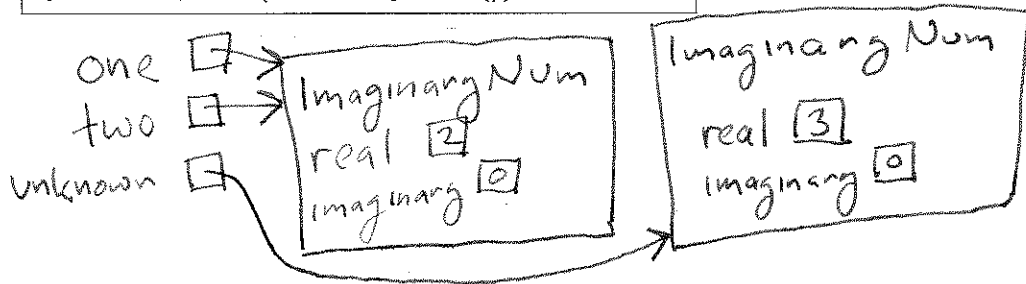
```
int val = 7;
ImaginaryNum num = new ImaginaryNum(val, val);
val = 8;
System.out.println(num.toString());
```

(a) Output: r = 7 i = 7
Diagram:



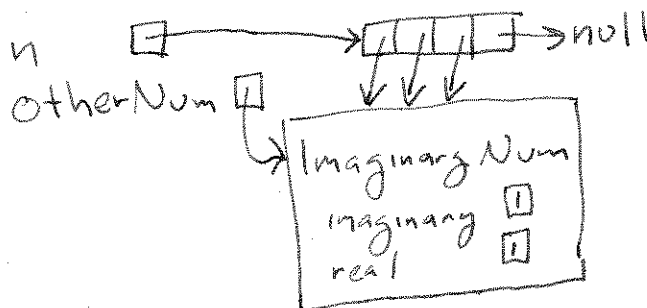
```
ImaginaryNum one = new ImaginaryNum(1,0);
ImaginaryNum two = one;
two.setReal(2);
ImaginaryNum unknown = two.addAndCreate(1);
System.out.println(unknown.getReal());
```

(b) Output: 3
Diagram:



```
ImaginaryNum[] n = new ImaginaryNum[4];
ImaginaryNum otherNum = new ImaginaryNum(1,1);
for(int i = 0; i < 3; i++) {
    n[i] = otherNum;
}
System.out.println(n[3]);
```

(c) Output: null
Diagram:



2. (6 points) For each code snippet below predict its output. (You do *not* need to draw a diagram, but you may if it might help you.)

```
String[] data = new String[10];  
data[0] = "Robot";  
data[1] = "Pirate";  
data[2] = "Ninja";  
System.out.println(data.length);
```

(a) Output: 10

```
double percent = 0.70;  
int intPercent = (int) percent;  
double result = (double) (100 * intPercent);  
System.out.println(result);
```

(b) Output: 0

```
ArrayList<String> al = new ArrayList<String>();  
al.add("Hello");  
al.add("World");  
System.out.println(al.get(1));
```

(c) Output: World

3. (8 points) For each for loop below, write down how many times its body will execute, or indicate that we can't tell from the information given.

```
for (int i = 0; i <= 6; i++) {  
    // loop body elided  
}
```

(a) Answer: 7

```
int maxTimer = 10;  
while (maxTimer > 3) {  
    maxTimer--;  
}
```

(b) Answer: 7

```
int[] data = new int[10];  
for(int data2 : data) {  
    // loop body elided  
}
```

(c) Answer: 10

```
while(true) {  
    System.out.println("in loop body");  
    break;  
}
```

(c) Answer: 1

4. (5 points) Explain the difference in behavior between the following two code examples (if any). If they are functionally different, explain why they are different AND which one you would prefer. If they are functionally the same, just write "They are the same."

```
System.out.println("Password?");
Scanner in = new Scanner(System.in);
String secretWord = "secret";
if(secretWord.equals(in.next())) {
    System.out.println("You won!");
}
```

and

```
System.out.println("Password?");
Scanner in = new Scanner(System.in);
String secretWord = "secret";
if(secretWord == in.next()) {
    System.out.println("You won!");
}
```

They are different. `==` determines if they are the same object, `equals` determines if they have the same contents. Only `equals` will give you correct behavior.

5. (6 points) For each of the code snippets below, there is a bug that causes the code to work incorrectly (i.e. the problem is just not weird style). First circle the bug, then write code that fixes the problem (if only one line is wrong, you only have to rewrite that one line).

```
ImaginaryNum seven;
seven.setReal(49.0/7);
```

Circle the bug in the code above and write code that fixes the problem:

`ImaginaryNum seven = new ImaginaryNum(0,0);`

```
String miss = "Mississippi";
miss.replace("i", "*");
System.out.printf("Mississippi without 'i's: %s\n", miss);
```

Circle the bug in the code above and write code that fixes the problem:

`miss = miss.replace("i", "*");`

6. (6 points) *Unit Testing*: Consider the documentation for the method below.

```
/**
 * Repeats the given string a given number of times
 *
 * For example, repeatString("hello",3) returns "hellohellohello"
 *
 * @param input
 * @param timesToRepeat
 * @return input string repeated timesToRepeat times
 */
public static String repeatString(String input, int timesToRepeat) {
    // body code elided
}
```

In the table below, give three sets of values that would constitute a good test set for this method. For each argument you list, say briefly why it should be in the unit test:

Argument to method	Expected Result	Why is this a good test?
a) "hello", 3	"hellohellohello"	correct result easy to figure out
b) "hello", 0	""	Bounding case when timesToRepeat = 0
c) "", 100	""	Bounding case when string is empty

Cases with special characters or long inputs might also be good