1. Write a string equality method checking for exact string equality given the following method header:

```
public static boolean stringEquals(String s1, String s2)
```

2. Recursively define the Fibanacci sequence, then write a recursive solution. Then, write an iterative solution. Example of Fibanacci sequence: 1 1 2 3 5 8 13 21 … (The first 2 numbers are always 1)

3. Given an xml tree, define a recursive print algorithm that will print all labels for each node with the proper indentation level for each node’s depth. In the contract, the int tab is how many spaces to print out before the label.

```
public static void printXMLLabels(XMLTree xml, int tab,
SimpleWriter out)
```

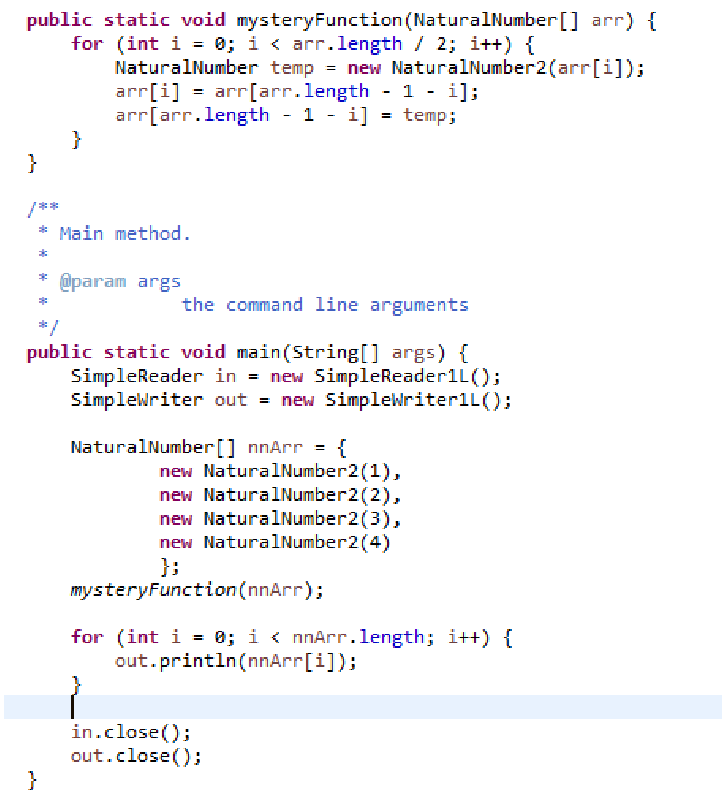
4. Write a recursive division by 2 method for NaturalNumber

```
public static void divBy2(NaturalNumber)
```

5. Write the fast powering method for NaturalNumber

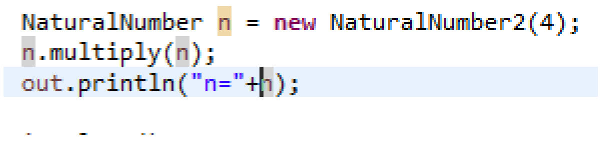
```
public static NaturalNumber fastPower(NaturalNumber n, int p)
```

6. Trace and draw the following code using reference diagrams as shown in class. What does it print in the end?



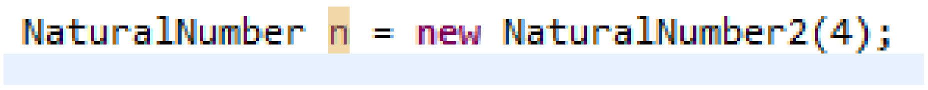
```
public static void mysteryFunction(NaturalNumber[] arr) {
    for (int i = 0; i < arr.length / 2; i++) {
        NaturalNumber temp = new NaturalNumber2(arr[i]);
        arr[i] = arr[arr.length - 1 - i];
        arr[arr.length - 1 - i] = temp;
    }
}

/**
 * Main method.
 *
 * @param args
 *            the command line arguments
 */
public static void main(String[] args) {
    SimpleReader in = new SimpleReader1L();
    SimpleWriter out = new SimpleWriter1L();

    NaturalNumber[] nnArr = {
            new NaturalNumber2(1),
            new NaturalNumber2(2),
            new NaturalNumber2(3),
            new NaturalNumber2(4)
            };
    mysteryFunction(nnArr);

    for (int i = 0; i < nnArr.length; i++) {
        out.println(nnArr[i]);
    }

    in.close();
    out.close();
}
```

7. What can the client always assume about a method using design by contract principles?
    a. That the result will be non-null
    b. The ensures clause will hold
    c. The method will be efficient
    d. None of the above
8. What are the three types of test casts that are useful to look for when creating a test plan?
9. What defines a recursive function?
    a. They are methods with large stacks
    b. They are like iterative solutions but always more efficient
    c. They are methods that call themselves
    d. They are methods that overwrite their own memory
10. In your own words, what is the benefit to binary search/interval halving?
11. What is the expected outcome of the following code:



```
NaturalNumber n = new NaturalNumber2(4);
n.multiply(n);
out.println("n="+n);
```

    a. n=16
    b. n=4
    c. Program crash
    d. None of the above
12. Fill in the blank: Arrays are _________ types that can hold a list of ________ type.
    a. Primitive, primitive
    b. Primitive, reference
    c. Reference, primitive
    d. Reference, reference
    e. Reference, any
    f. Any, reference
    g. Primitive, any
    h. Any, primitive
13. In the following code, ________ is the static/declared type, and ______ is the dynamic/object type.



```
NaturalNumber n = new NaturalNumber2(4);
```

    a. NaturalNumber, NaturalNumber
    b. NaturalNumber, NaturalNumber2
    c. NaturalNumber2, NaturalNumber
    d. NaturalNumber2, NaturalNumber2

ANSWERS:

1. Write a string equality method checking for exact string equality given the following method header:

```java
public static boolean stringEquals(String s1, String s2) {
    boolean eq = s1.length() == s2.length();
    int i = 0;
    while (eq && i < s1.length()) {
        eq = s1.charAt(i) == s2.charAt(i);
        i++;
    }
    return eq;
}
```

2. Fib(n) = Fib(n-1) + Fib(n-2)

```java
public static int fib(int n) {
    int result = 1;
    if (n > 1) {
        result = fib(n - 1) + fib(n - 2);
    }
    return result;
}

public static int fibIterative(int n) {
    int last2[] = { 1, 1 };
    for (int i = 1; i < n; i++) {
        int temp = last2[0] + last2[1];
        last2[0] = last2[1];
        last2[1] = temp;
    }
    return last2[1];
}
```

3.

```java
public static void printXMLLabels(XMLTree xml, int tab, SimpleWriter out) {
    for (int i = 0; i < tab; i++) {
        out.print(" ");
    }
    out.println(xml.label());
    if (xml.isTag()) {
        for (int i = 0; i < xml.numberOfChildren(); i++) {
            printXMLLabels(xml.child(i), tab + 4, out);
        }
    }
}
```

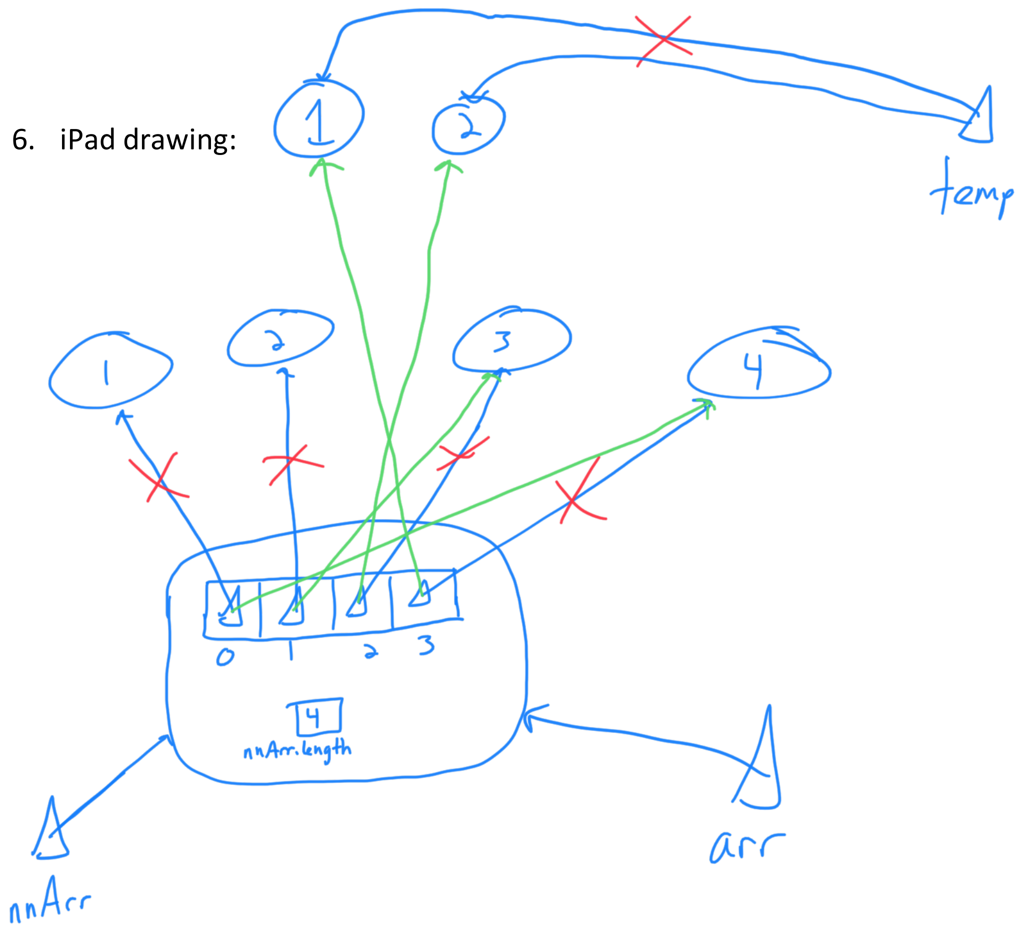
4.

```
public static void divBy2(NaturalNumber n) {
    final NaturalNumber ten = new NaturalNumber2(10);
    if (n.compareTo(ten) >= 0) {
        int lastDig = n.divideBy10();
        int secondLastDig = n.divideBy10();
        lastDig /= 2;
        if (secondLastDig % 2 != 0) {
            lastDig += 5;
        }
        n.multiplyBy10(secondLastDig);
        ;
        divBy2(n);
        n.multiplyBy10(lastDig);
    } else {
        int lastDig = n.divideBy10();
        n.multiplyBy10(lastDig / 2);
    }
}
```

5.

```
public static NaturalNumber fastPower(NaturalNumber n, int p) {
    NaturalNumber result = new NaturalNumber2(1);
    if (p > 0) {
        result = fastPower(n, p / 2);
        NaturalNumber result2 = new NaturalNumber2(result);
        result.multiply(result2);
        if (p % 2 != 0) {
            result.multiply(n);
        }
    }
    return result;
}
```

6. iPad drawing:



Final print

4
3
2
1

mysteryMethod reverses all the pointers, but creates new NNs for half of arr.

7. D. The client can only assume the postcondition/ensures clause IF they follow the precondition/requires clause. That is, they cannot "always" assume it is true.

8. Routine cases, challenging cases, and edge cases

9. C

10. Binary search and interval halving effectively cuts down the possible solutions by ½ every update. This is beneficial because most other solutions can only discard one entry each time, but interval halving lets us discard half of our entire set.

11. D. The actual outcome is 0, but that doesn't make any sense. Postcondition for multiply ensures that this is updated to this = #this * n, but it also says that n is restored. Since this = n, we don't know which one will hold.

12. E. Arrays are reference types that can hold lists of any type in all of Java.

13. B, NaturalNumber is an interface, and is the static/declared type here. NaturalNumber2 is the class, and is the dynamic/object type.