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

public static boolean stringEquals(String s1, String s2)

1. 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)
2. 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)

1. Write a recursive division by 2 method for NaturalNumber

public static void divBy2(NaturalNumber)

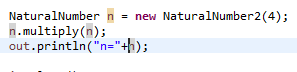
1. Write the fast powering method for NaturalNumber

public static NaturalNumber fastPower(NaturalNumber n, int p)

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



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



* 1. n=16
  2. n=4
  3. Program crash
  4. None of the above

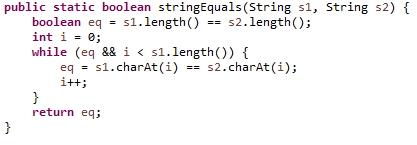
1. Fill in the blank: Arrays are \_\_\_\_\_\_\_\_ types that can hold a list of \_\_\_\_\_\_\_ type.
   1. Primitive, primitive
   2. Primitive, reference
   3. Reference, primitive
   4. Reference, reference
   5. Reference, any
   6. Any, reference
   7. Primitive, any
   8. Any, primitive
2. In the following code, \_\_\_\_\_\_\_ is the static/declared type, and \_\_\_\_\_\_ is the dynamic/object type.



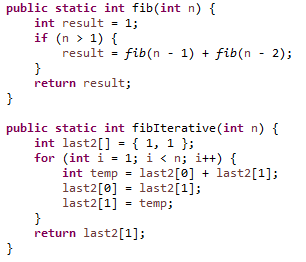
* 1. NaturalNumber, NaturalNumber
  2. NaturalNumber, NaturalNumber2
  3. NaturalNumber2, NaturalNumber
  4. NaturalNumber2, NaturalNumber2

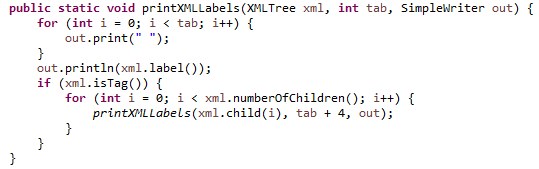
**ANSWERS**:

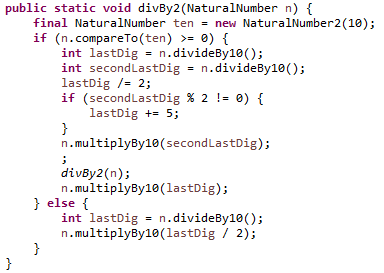
1. Write a string equality method checking for exact string equality given the following method header:

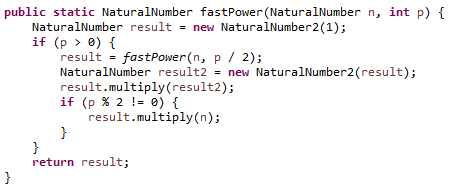


1. Fib(n) = Fib(n-1) + Fib(n-2)









1. iPad drawing:
2. 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.
3. Routine cases, challenging cases, and edge cases
4. C
5. 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.
6. 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.
7. E. Arrays are reference types that can hold lists of any type in all of Java.
8. B, NaturalNumber is an interface, and is the static/declared type here. NaturalNumber2 is the class, and is the dynamic/object type.