CSE 2221 - Midterm 2 Study Guide

Trees

1. What is the definition of the size of a tree?

# of nodes in a tree

2. What is the definition of the height of a tree?

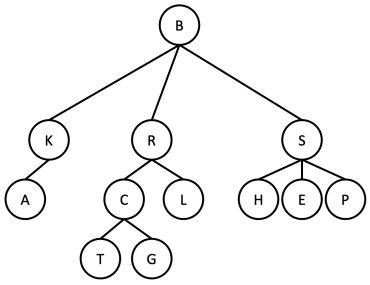
longest path from root to leaf in a tree (largest # of nodes)

3. What is the definition of a path? What is the length of a path?

a series of nodes connected by edges where each node is a parent or child of another

length is the # of nodes in a path

Consider the following tree for questions 5-11:



5. What is the root of the tree?

B

6. What are the names of the children of the root node?

K R S

7. How many children does node R have?

2

8. How many leaves are in the tree?

7

9. How many children does node E have?

∅

10. What node is the parent of A?

K

11. What is the size and height of the tree?

12 4

Not given as questions, but you need to know XMLTree functions such as numberOfChildren, isTag, child, label, etc.

NaturalNumbers

13. What is the value range of a NaturalNumber? Give it in notation using square brackets and parenthesis.

[0, ∞) technically limited by memory though

14. What is the benefit of using the String constructor over the int constructor?

int has a limit while String has a much, much larger limit

15. What types of methods are in the NaturalNumber interface(s)?

instance methods

16. If I write "nn.isZero()", what is "nn" called?

receiver

17. What is the name of "nn" inside the "isZero" method?

this, "distinguished formal parameter"

18. What does "@ensures this = #this + 1" mean in a method contract?

new value of this will be equal to the old value of this plus one

19. If I write "nn.increment()", does anything change about "nn"? What changes?

yes, object value changes

20. When should you use copyFrom over transferFrom?

if you have x.copyFrom(y), this says you will still need y at its current value later in the code

21. If n1 > n2, what is the result of "n1.compareTo(n2)"?

any value x where x > 0 (i.e. a positive number)

22. In the function call "nn.multiplyBy10(k)", what is "k" for?

adding to ones place after multiplying

23. What are the kernel methods?

isZero, multiplyBy10, divideBy10 - very essential methods to be used in all NaturalNumber objects

24. What is the default value of a NaturalNumber?

∅

25. What is the resulting object value of the argument passed to a transferFrom call?

∅

References

26. What are reference types associated with?

objects

27. Is String a primitive or reference type?

reference

28. Is the value of a primitive type the actual value or is it a memory address?

actual

29. Is the value of a reference type the actual value or is it a memory address?

memory

30. How many values does a reference type have? What are they?

2 - reference value, object value

31. Draw the representation for the line "String myStr = "No class until March 30th lol";".

32. Draw the representation for the line "NaturalNumber myNN = new NaturalNumber2(400 + 55);"

33. What is the object value of the line of code from question 32?

455

34. Do we care what the reference value is? Why or why not?

No, we can't do arithmetic on the memory addresses, therefore they are arbitrary

35. What happens when you do "nn1 = nn2"? Assuming nn1 and nn2 are two different NaturalNumbers.

nn1 & nn2 will have the same reference value and therefore will be aliases

36. Is writing "a = 5, b = 5" in a tracing table the same as writing "a -> 5, b -> 5"? Assuming a and b are NaturalNumbers.

No, "a = 5, b = 5" can mean aliases or two different reference values with the same object value and "a → 5, b → 5" strictly means the latter

37. What gets copied in a copyFrom call?

the object value

38. What does "str2 = str1 + "ing";" do to the object value of "str1"? What about "str2"? Assuming "str1" and "str2" are two already initialized Strings. Why can we change the object value of Strings this

way, but must call methods for NaturalNumbers?

str1 doesn't change, str2 does change
b/c Strings are built directly into Java

39. What is an alias? Draw the representation using NaturalNumbers. What does this look like in a tracing table?

when two variables share the same reference value

n1 = 75
n2 = 75
} potentially incorrect, ambiguous

n1, n2 → 75 ✓

40. What is an immutable type?

a reference type where the object value cannot be changed without changing the reference value
or the reference type contains no methods that can directly change the object value without affecting the reference value

41. What is a mutable type? Don't just say the opposite of an immutable type.

a reference type where the object value can be changed without changing the reference value

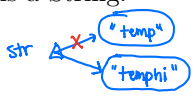
42. What type is a String?

immutable

43. What type is a NaturalNumber?

mutable

44. What happens to the reference ~~type~~ value of "str" when we execute the line "str += "hi";"? Assuming "str" is a String.



the reference value changes to a new memory address b/c immutable

45. What happens to the reference ~~type~~ value of "nn" when we execute the line "nn.divideBy10();"? Assuming "nn" is a NaturalNumber.

nothing

46. How are parameters passed for primitive types?

pass by value
copy by value

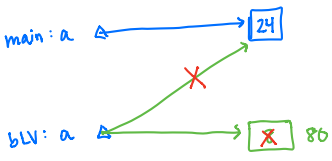
47. How are parameters passed for reference types?

pass by reference value
copy by reference value

48. How is "a" affected in the following method?

```
private static void budLightVirus(NaturalNumber a) {
     a = new NaturalNumber2(8);
     a.multiplyBy10();
}

public static void main(String[] args) {
     NaturalNumber a = new NaturalNumber2(24);
     budLightVirus(a);
}
```



a doesn't change in main, unaffected

49. What is the correct way to compare two Strings? What about NaturalNumbers?

str1.equals(str2) nn1.compareTo(nn2) {>, <, >=, <=} some value

50. What happens if you do "n1 == n2"? What about "n1.equals(n2)"? Assuming n1 and n2 are NaturalNumbers.

both compare reference values

Arrays

51. Is an array primitive or reference?

reference

52. What types of values can an array hold?

any, as long as same type

53. What does "=" do for arrays?

copy the reference value

54. What does "==" do for arrays?

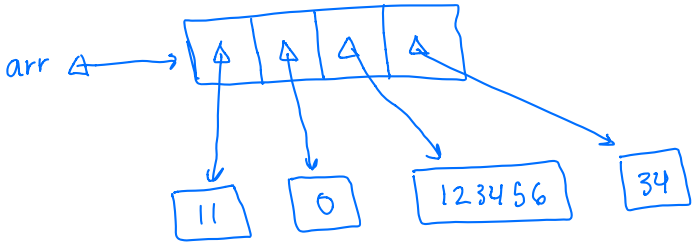
compare reference values

55. How are arrays passed to methods?

pass by reference value

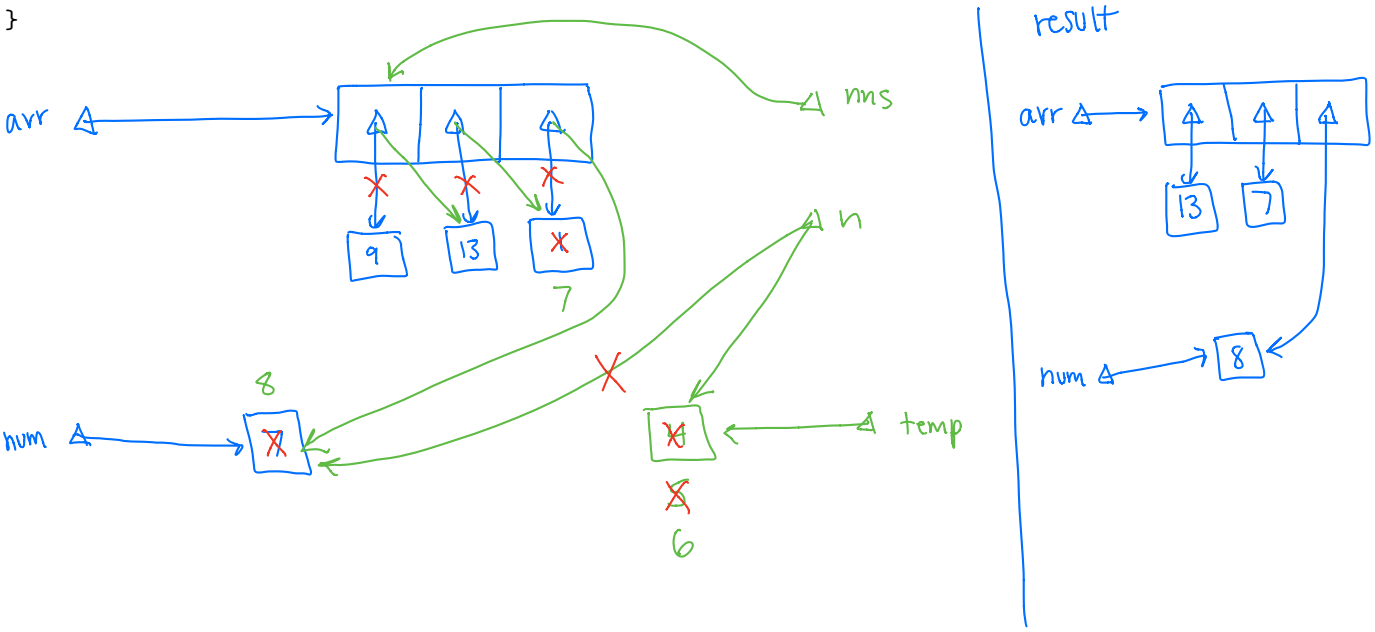
56. Draw the representation for an array of random ints (length = 5).

57. Draw the representation for an array of random NaturalNumbers (length = 4).



58. Draw the representation of the variables and trace them through the code below:

```
private static void badCoronavirusJoke(NaturalNumber n, NaturalNumber[] nns) {
     NaturalNumber temp = new NaturalNumber2(4);
     nns[0] = nns[1];
     nns[1] = nns[2];
     nns[2] = n;
     n.increment();
     n = temp;
     n.increment();
     temp.increment();
     nns[1].add(temp);
}
public static void main(String[] args) {
     NaturalNumber[] arr = new NaturalNumber2[3];
     arr[0] = new NaturalNumber2(9);
     arr[1] = new NaturalNumber2(13);
     arr[2] = new NaturalNumber2(1);
     NaturalNumber num = new NaturalNumber2(7);
     badCoronavirusJoke(num, arr);
}
```



Contracts

59. Assuming nn1 is a NaturalNumber, what value am I referring to when I say "nn1 = #nn1 + 23" in a method contract?

object value

60. Can a reference parameter by null? If yes, do we allow it in this class?

yes, no

61. What would a restores mode parameter look like in a method contract?

@restores var

62. What would a clears mode parameter look like in a method contract?

@ clears var

63. What would a replaces mode parameter look like in a method contract?

@ replaces var
@ ensures var = 5

64. What would an updates mode parameter look like in a method contract?

@ updates var
@ ensures var = #var + 5

65. What is the default parameter mode?

restores

66. What is the relationship between the input parameter and a replaces mode parameter?

no relationship

67. When we use a parameter name in a method contract, what value are we referring to when the parameter is a (a) primitive type or (b) reference type?

THE value object value

68. Do we allow the following statement? "nn.multiply(nn);"

no this ≠ arg

Aliasing

69. What is the simplest way to create an alias?

nn1 = nn2;

70. Can you ever have harmful aliasing in a no argument instance method?

technically yes if you do: NaturalNumber n = this;

71. How many "points of contact" do you have in harmless aliasing? i.e. how many reference variables reference an object.

1

72. How many "points of contact" do you have in harmful aliasing?

2+

Mathematical String Notation

73. Can you have duplicate entries in a string? Does order of the entries matter?

yes yes

74. Write the mathematical string notation for programming type String equal to "CSE2221". What is the mathematical type of this?

⟨'C', 'S', 'E', '2', '2', '2', '1'⟩ String of characters

75. Write the mathematical string notation for programming type int equal to [1, 6). What is the mathematical type of this?

⟨1, 2, 3, 4, 5⟩ String of integers

76. What operator do you use to concatenate two programming Strings? What about two mathematical strings?

programming → +

math → *

77. What is the definition of a substring? Give an example.

if A is a substring of B, all elements of A occur in B

⟨1, 2⟩ → ⟨7, 1, 2, 4⟩

78. What is the definition of a prefix? Give an example.

all elements of A occur first in B

⟨1, 2⟩ → ⟨1, 2, 7, 6, 11⟩

79. What is the definition of a suffix? Give an example.

all elements of A occur last in B

⟨1, 2⟩ → ⟨8, 14, 1, 2⟩

80. What is "s[1, 7)" of "s = "Go Buckeyes!""?

"o Buck"

Recursion Part 1

81. Simply put, what is recursion?

a function that calls itself

82. What must be done in order for recursion to work? Hint: Can I reverse "Hello" right now? No, so what must I do?

You must pass a smaller subproblem to successive method calls

83. We keep doing recursive steps until we reach what? What is the significance of this point that we have reached?

base case, it is the smallest subproblem we can solve

84. What is the basic structure for a recursive method?

```
if (base case condition) {
        return some basic value
} else {
        call method again w/ smaller subproblem
}
```

this can always differ, but this is just a classic structure for a recursive algorithm

85. How do we trace over a recursive call? How do we determine the behavior of the call?

Treat as one atomic step (i.e. step over in debug)

Based on method contract

86. How many return statements should we have in our recursive method?

1

87. Considering the slow powering algorithm, what is the base case? What is the condition for a recursive step? What is our subproblem?

① exponent == 0, return 1

② exponent > 0

③ $n^{exponent-1}$, power(n, exponent - 1)

Object Oriented Programming

88. Fill in the blank. A Class implements an Interface.

89. Fill in the blank. A Class extends a Class.

90. Fill in the blank. An Interface extends an Interface.

91. If a Class implements an Interface, how many of the methods from the Interface are implemented in the Class?

all

92. How can a Class call a method from its parent Class? What is the keyword?

super

93. If a Class extends a Class, how many of the methods from the parent Class are overrode in the child Class?

[∅, all]

94. How do you override a method in a child Class?

@Override above method signature

95. If you have overrode a method in a child Class, can you call the original method from the parent Class?

yes, with super

96. What is overloading? Give an example.

2+ methods with same method name but different return types and/or parameter types and/or number of parameters

97. Consider the following line: "SimpleWriter out = new SimpleWriter1L("myFile.txt");". What is the declared type? What is the object type?

SimpleWriter, SimpleWriter1L

98. Consider the following line: "NaturalNumber nn = new NaturalNumber2(new NaturalNumber1L(10));". What is the declared type? What is the object type?

NaturalNumber, NaturalNumber2

99. What is best practice regarding the declared type?

have it be an interface

100. What is the name of the property of Object Oriented Programming that determines which instance method (from what class) is called?

polymorphism

101. Can an instance method be a void return type? What are 3 examples of such methods.

yes; add, subtract, & multiply for NNs

Recursion Part Two (Including Recursion on Trees)

102. What do we call the approach for determining if a recursive method is correct or not?

confidence - building

103. What is the metric used for in a recursion?

determining if we can solve the problem or need to create a subproblem

104. If there exists a 2nd child of the root node, what can we say about this child?

it is a subtree

105. Write a recursive function that calculates the number of edges in a tree.

```
private static int numEdges(XMLTree tree) {
    int num = 0;
    if (tree.numberOfChildren() > 0) {
        int childIndex = 0;
        while (childIndex < tree.numberOfChildren()) {
            num += (1 + numEdges(tree.child(childIndex)));
            childIndex++;
        }
    }
    return num;
}
```

106. Write a recursive function that determines how many nodes in the tree have an even number of children.

```
private static int numEvenChildren(XMLTree tree) {
    int num = 0;
    if (tree.numberOfChildren() % 2 == 0) {
        num++;
    }
    int childIndex = 0;
    while (childIndex < tree.numberOfChildren()) {
        num += numEvenChildren(tree.child(childIndex));
        childIndex++;
    }
    return num;
}
```

107. Write a recursive function that searches for the first occurrence of a node with n number of children.

```
private static boolean search(XMLTree tree, String label) {
    boolean found = false;
    if (tree.label().equals(label)) {
        found = true;
    } else {
        int childIndex = 0;
        while (childIndex < tree.numberOfChildren() && !found) {
            found = search(tree.child(childIndex), label);
            childIndex++;
        }
    }
    return found;
}
```

108. What is an issue with this recursive function? Besides there being multiple returns.

```
private static int factorial(int n) {
     if (n == 3) {
      return 6;
     } else {
          return factorial(n-1) * n;
     }
}
```

we disallow the caller to compute 2! or 1!

results in infinite recursion... how?

109. What is an issue with this recursive function? Besides there being multiple returns.

```
private static int fibonacci(int n) {
     if (n == 1) {
      return 1;
     } else {
          return fibonacci(n-1) + fibonacci(n-2);
     }
}
```

if n == 3, then return fib(2) + fib(1)

↓

this will try to compute fib(1) + fib(0)

↓

doesn't exist

∞ recursion

Testing

110. What type of testing are we performing in this class?

Unit testing

111. What type of testing occurs when we try to combine multiple components into a system?

integration testing

112. What type of testing occurs when we try to test an entire system?

system testing

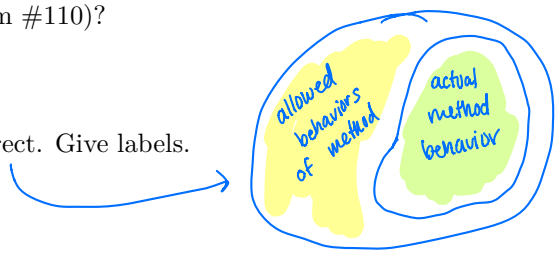
113. (NOT IN SLIDES) What type of testing occurs when we try to make sure a new features has not broken old, previously working features?

regression testing

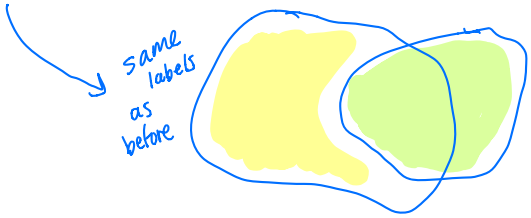
114. What do we call the item being tested (using the method from #110)?

unit under test

115. Draw a diagram for what it looks like for a method to be correct. Give labels.



116. Draw a diagram for what it looks like for a method to be incorrect. Give labels.



117. What are we trying to show in testing?

a method is incorrect

118. What is the difference between testing and debugging?

testing → no known bugs
run entire method
evaluate output

debugging → known bug
step through each line of code
evaluate intermediate states

119. If we wrote 1000 test cases for a function and all test cases pass, is the function guaranteed to be correct?

no

120. What do we call a set of test cases for one method?

test plan / fixture

Consider a function "toInt()" that is overloaded to take ints, chars, doubles, and booleans and simply returns the integer representation of the input (keep in mind, chars are just 8 bit numbers deep down and booleans are just 0 or 1). The only precondition is it must be a non-negative input

121. Write 12 different good test cases for the above function (only write the function name with the input argument). Write whether they are "routine" with R, "boundary" with B, or "challenging" with C. Hint: don't have all routine test cases. You may NOT copy the ones from future questions.

toInt(0) B
toInt(17) R

---

toInt('0') C
toInt('A') R/C
toInt('%') R/C

---

toInt(true) R/C
toInt(false) R/C

---

toInt(0.0000001) B/C
toInt(1.0000000) R
toInt(0.9999999999999999) R/C
toInt(3.14159) R
toInt((double) Math.pow(2,31)) + 10) B/R/C

122. Is toInt(-1) a good test case? Why or why not?

no, we don't allow negative inputs

123. Is toInt("a") a good test case? Why or why not?

no, we don't have a method for strings

124. Is toInt(0.00000) a good test case? Why or why not?

yes, boundary

125. Is toInt(Math.pow(2, 31)) a good test case? Why or why not?

yes, boundary

126. Is toInt('5') a good test case? Why or why not?

yes, routine, but will it return 5 or correct value of 53 (look at ASCII table)

127. Is toInt(new NaturalNumber2(0)) a good test case? Why or why not?

no, we don't have a method for NNs

128. Write actually JUnit test cases (like ones you would write in code) for 2 of your answers from #121. What should be above your test case? What do you need in the test case to tell the program it passed?

```
@Test
public static void testToIntWithNumericalChar(){
        char input = '0';
        int expected = 48;
        int actual = toInt(input);
        AssertEquals(expected, actual);
}
```

```
@Test
public static void testToIntWithFalseBoolean(){
        boolean input = false;
        int expected = 0;
        int actual = toInt(input);
        AssertEquals(expected, actual);
}
```