1.1

Explain why a method that does not match its declaration in the interface would not be discovered during white-box testing.

Methods which do not match their declaration in the interface are detected by the Java compiler, and thus would never get to white-box testing.

2.1

List two boundary conditions that should be checked when testing method readInt below.

/\*\* *Returns an integer data value within range minN and maxN inclusive*

\* @param *scan a Scanner object*

\* @param *minN smallest possible value to return*

\* @param *maxN largest possible value to return*

\* @return *the first value read between minN and maxN*

\*/

public static int readInt(Scanner scan, int minN, int maxN) {

if (minN > maxN) {

throw new IllegalArgumentException(

"In readInt, minN " + minN

+ " not <= maxN " + maxN);

}

boolean inRange = false; // Assume no valid number read.

int n = 0;

while (!inRange) { // Repeat until valid number read.

System.out.println("Enter an integer from " + minN + " to "

+ maxN + ": ") ;

try {

n = scan.nextlnt();

inRange = (minN <= n & & n <= maxN) ;

} catch (inputMismatchException ex) {

scan.nextLine();

System.out.println("not an integer ‐ try again");

}

} // End while

return n; // n is in range

}

minN == maxN (e.g. minN = maxN = 1), and test for input 1 and 2

minN > maxN (e.g. minN = 5 and maxN = 2)

3.1

Explain why a method that does not match its declaration in the interface would not be

discovered during white-box testing.

A main method can be used as a driver to test the methods of a class. It can’t be used as a stub because a main method is a starting point for class execution.

4.1

Modify the test(s) in the list for Example 3.3 to verify a method that finds the last occurrence

of a target element in an array?

The search method discussed in Section 3.2 finds the first occurrence of a target in an array. To find the last occurrence of a target in an array, we need to change the test:

There is more than one occurrence of the target element and we find the first occurrence.

to

There is more than one occurrence of the target element and we find the last occurrence.

5.1

Why did the first version of method search that passed the first test itemNotFirstElementInSingleElementArray

contain only the statement return −1?

It’s only requirement was to pass the test that returned the correct answer (-1) when the target was not present in a 1-element array.

6.1

Explain why it is not necessary to write a test to verify that readInt works properly when

the input consists of an invalid integer string, followed by an out-of-range integer, then

followed by an integer that is in range.

You should have a test that verifies the integer result and expected prompt are correct when the input consists of an invalid integer string followed by an integer that is in range, and you should also have a test that verifies the result and prompt when the input consists of an out of range integer followed by an integer that is in range. So the test in question is not necessary because it would pass only if both of these other tests passed and fail if either of them failed.

7.1

The following method does not appear to be working properly if all data are negative

numbers. Explain where you would add diagnostic output statements to debug it, and give

an example of each statement.

/*\*\* Finds the target value in array elements* x[start] *through* x[last].

@param x *array whose largest value is found*

@param start *first subscript in range*

@param last *last subscript in range*

@return *the largest value of* x[start] *through* x[last]

@pre first <= last

*\**/

public int findMax(int[] x, int start, int last) {

if (start > last)

throw new IllegalArgumentException("Empty range");

int maxSoFar = 0;

for (int i = start; i < last; i++) {

if (x[i] > maxSoFar)

maxSoFar = i;

}

return maxSoFar;

}

You should add braces to the if statement and insert a debugging statement to show the value of i and the value of maxSoFar after each new assignment of value to maxSoFar.

{ maxSoFar = i;

System.out.println (“index i: i “ + i +” , “ + “maxSoFar: “ + maxSoFar);

}

I would also display the value of maxSoFar after loop exit. These statements should show that the value 0 is always returned for negative input. However, it should also show that the result is incorrect for positive data because the value returned would always be a subscript instead of a number.

7.3

Explain the rationale for the position of the breakpoints in method getSentence.

A breakpoint should be set just at entry to the while loop (line 21) and at the return statement just after loop exit (line 28). This allows you to see the value of sentence before and after loop execution.

-