Due to the print book page limit, we cannot inlcude all good CheckPoint questions in the physical book. The CheckPoint on this Website may contain extra questions not printed in the book. The questions in some sections may have been reordered as a result. Nevertheless, it is easy to find the CheckPoint questions in the book on this Website. Please send suggestions and errata to Dr. Liang at y.daniel.liang@gmail.com. Indicate the book, edition, and question number in your email. Thanks!

Chapter 18 Check Point Questions

Section 18.2

▼18.2.1

What is a recursive method? What is an infinite recursion?

A recursive method is the one that calls itself. An infinite recursion is the one that never stops.

Hide Answer

▼18.2.2

How many times is the factorial method in Listing 18.1 invoked for factorial(6)?

six times. (base case factorial(0))

Hide Answer

▼18.2.3

Show the output of the following programs and identify base cases and recursive calls.

```
(a)
public class Test {
  public static void main(String[] args) {
    System.out.println(
      "Sum is " + xMethod(5));
  }

  public static int xMethod(int n) {
    if (n == 1)
      return 1;
    else
      return n + xMethod(n - 1);
  }
}

(b)
public class Test {
  public static void main(String[] args) {
    xMethod(1234567);
  }

  public static void xMethod(int n) {
    if (n > 0) {
      System.out.print(n % 10);
      xMethod(n / 10);
    }
  }
}
```

(a) Sum is 15 (5 + 4 + 3 + 2 + 1 = 15)
(b) 7654321

Hide Answer

▼18.2.4

Write a recursive mathematical definition for computing 2^n for a positive integer n.

```
f(n) = 2 if n = 1
f(n) = 2 * 2^(n-1) for (n > 1)
```

Hide Answer

▼18.2.5

Write a recursive mathematical definition for computing x^n for a positive integer n and a real number x.

```
f(n) = x if n = 1
f(n) = x * x^(n-1) for (n > 1)
```

Hide Answer

▼18.2.6

Write a recursive mathematical definition for computing 1 + 2 + 3 + ... + n for a positive integer n.

```
f(n) = 1 if n = 1
f(n) = f(n-1) + n for (n > 1)
```

Hide Answer

Section 18.3

▼18.3.1

Show the output of the following two programs:

```
(a)
public class Test {
  public static void main(String[] args) {
    xMethod(5);
  }

  public static void xMethod(int n) {
    if (n > 0) {
      System.out.print(n + " ");
      xMethod(n - 1);
    }
  }
}

(b)
public class Test {
  public static void main(String[] args) {
    xMethod(5);
  }

  public static void xMethod(int n) {
    if (n > 0) {
      xMethod(n - 1);
      System.out.print(n + " ");
    }
  }
}
```

(a) The output is 5 4 3 2 1
(b) The output is 1 2 3 4 5

Hide Answer

▼18.3.2

What is wrong in the following method?

```
(a)
public class Test {
  public static void main(String[] args) {
    xMethod(1234567);
  }

  public static void xMethod(double n) {
    if (n != 0) {
      System.out.print(n);
      xMethod(n / 10);
    }
  }
```

```
}

(b)
public class Test {
  public static void main(String[] args) {
    Test test = new Test();
    System.out.println(test.toString());
  }

  public Test() {
    Test test = new Test();
  }
}
```

(a) n is double. There is no guarantee that n != 0 will be eventually false.
(b) Infinite recursion due to new Test() inside the constructor Test().

Hide Answer

▼18.3.3

How many times is the fib method in Listing 18.2 invoked for fib(6)?

25 times (Why?
number of time fib is invoked in fib(0) =
1
number of time fib is invoked in fib(1) =
1
number of time fib is invoked in fib(2) =
1+ number of time fib is invoked in fib(1)+number of time fib is invoked in fib(2) =1+1+1=3
number of time fib is invoked in fib(3) =
1+ number of time fib is invoked in fib(1)+number of time fib is invoked in fib(2) = 1+1+3=5
number of time fib is invoked in fib(4) =
1+ number of time fib is invoked in fib(2)+number of time fib is invoked in fib(3) = 1+3+5=9
number of time fib is invoked in fib(5) =
1+ number of time fib is invoked in fib(3)+number of time fib is invoked in fib(4) = 1+5+9=15
number of time fib is invoked in fib(6) =
1+ number of time fib is invoked in fib(4)+number of time fib is invoked in fib(5) = 1+9+15=25

Hide Answer

Section 18.4

▼18.4.1

Describe the characteristics of recursive methods.

One or more base cases (the simplest case) are used to stop recursion. Every recursive call reduces the original problem, bringing it increasingly close to a base case until it becomes that case.

Hide Answer

▼18.4.2

For the isPalindrome method in Listing 18.3, what are the base cases? How many times is this method called when invoking isPalindrome("abdxcxdba")?

The base cases are (1) s.length() <= 1 and (2) s.charAt(0) != s.charAt(s.length - 1)
When invoking isPalindrome("abdxcxdba"), the isPalindrome method is called 5 times.

Hide Answer

▼**18.4.3**
Show the call stack for isPalindrome("abcba") using the method defined in Listing 18.3.

Omitted

Hide Answer

Section 18.5

▼**18.5.1**
Show the call stack for isPalindrome("abcba") using the method defined in Listing 18.4.

Omitted

Hide Answer

▼**18.5.2**
Show the call stack for selectionSort(new double[]{2, 3, 5, 1}) using the method defined in Listing 18.5.

Omitted

Hide Answer

▼**18.5.3**
What is a recursive helper method?

An overloaded method with additional parameters.

Hide Answer

Section 18.6

▼**18.6.1**
What is the base case for the getSize method?

The base case for the getSize(File d) method is that d is a file.

Hide Answer

▼**18.6.2**
How does the program get all files and directories under a given directory?

The program gets all files and directories under the directory d using d.listFiles(), which returns an array of File objects under the directory.

Hide Answer

▼**18.6.3**
How many times will the getSize method be invoked for a directory if the directory has three subdirectories and each subdirectory has four files?

4 times for the directories and 4 * 4 time for all the files. So, the total is 20.

Hide Answer

▼18.6.4

Will the program work if the directory is empty (i.e., it does not contain any files)?

Yes.

Hide Answer

▼18.6.5

Will the program work if line 20 is replaced by the following code?

```
for (int i = 0; i < files.length; i++)
```

No. The directory may be empty.

Hide Answer

▼18.6.6

Will the program work if lines 20-21 is replaced by the following code?

```
for (File file: files)
  size += getSize(file); // Recursive call
```

No. files may be null.

Hide Answer

Section 18.7

▼18.7.1

How many times is the moveDisks method in Listing 18.8 invoked for moveDisks(5, 'A', 'B', 'C')?

2^5 - 1

Hide Answer

Section 18.8

▼18.8.1

How do you obtain the midpoint between two points?

The midpoint between p1 and p2 is ((p1.x + p2.x)/2, (p1.y + p2.y)/2), which can be obtained by invoking p1.midpoint(p2).

Hide Answer

▼18.8.2

What is the base case for the displayTriangles method?

The base case for the displayTriangles method is order == 0.

Hide Answer

▼18.8.3

How many times is the displayTriangles method invoked for a Sierpinski triangle of order 0, order 1, order 2, and order n?

The displayTriangles method is invoked one time for order 1, 4 times for order 1, 1 + 3 * 3 times for order 2, and 1 + 3^n for order n.

Hide Answer

18.8.4

What happens if you enter a negative order? How do you fix this problem in the code?

Will be an infinite loop. To fix it, add if (order < 0) return in the beginning of the method displayTriangle.

Hide Answer

18.8.5

Instead of drawing a triangle using a polygon, rewrite the code to draw a triangle by drawing three lines to connect the points in lines 71-77.

Replace lines 71-77 with the following code:

```
// Draw a triangle to connect three points
Line line1 = new Line(p1.getX(), p1.getY(), p2.getX(), p2.getY());
Line line2 = new Line(p2.getX(), p2.getY(), p3.getX(), p3.getY());
Line line3 = new Line(p3.getX(), p3.getY(), p1.getX(), p1.getY());

this.getChildren().addAll(line1, line2, line3);
```

Hide Answer

Section 18.9

18.9.1

Which of the following statements are true?

a. Any recursive method can be converted into a nonrecursive method.
b. Recursive methods take more time and memory to execute than nonrecursive methods.
c. Recursive methods are always simpler than nonrecursive methods.
d. There is always a selection statement in a recursive method to check whether a base case is reached.

a. (TRUE)
b. (TRUE)
c. (FALSE)
d. (TRUE)

Hide Answer

18.9.2

What is a cause for a stack-overflow exception?

When a method is invoked, its contents are placed into a stack. If a method is recursively invoked, it is possible that the stack space is exhausted. This causes stack overflow.

Hide Answer

Section 18.10

▼18.10.1

Identify tail-recursive methods in this chapter.

The isPalindrome method in Listing 18.4, sort method in Listing 18.5, and binarySearch method in Listing 18.6 are tail-recursive.

Hide Answer

▼18.10.2

Rewrite the fib method in Listing 18.2 using tail recursion.

```
/** Return the Fibonacci number for the specified index */
public static long fib(long index) {
  return fib(index, 1, 0);
}

/** Auxiliary tail-recursive method for fib */
private static int fib(long index, int next, int result) {
  if (index == 0)
    return result;
  else
    return fib(index - 1, next + result, next);
}
```

Hide Answer