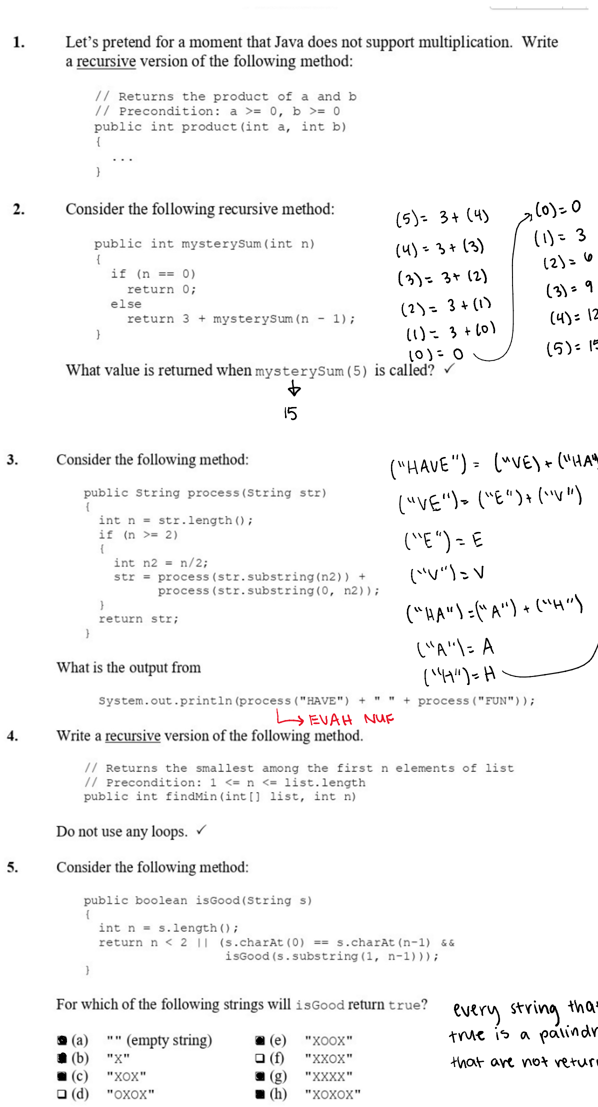
Ch 13 Homework Problems - Becca and Sean

Tuesday, January 17, 2023 12:09 PM



1. Let's pretend for a moment that Java does not support multiplication. Write a recursive version of the following method:

```
// Returns the product of a and b
// Precondition: a >= 0, b >= 0
public int product(int a, int b)
{
  ...
}
```

2. Consider the following recursive method:

```
public int mysterySum(int n)
{
  if (n == 0)
    return 0;
  else
    return 3 + mysterySum(n - 1);
}
```

(5)= 3+ (4)
(4) = 3 + (3)
(3)= 3+ (2)
(2)= 3 + (1)
(1)= 3 + (0)
(0)= 0

(0)= 0
(1)= 3
(2)= 6
(3)= 9
(4)= 12
(5)= 15

What value is returned when `mysterySum(5)` is called? ✓

15

3. Consider the following method:

```
public String process(String str)
{
  int n = str.length();
  if (n >= 2)
  {
    int n2 = n/2;
    str = process(str.substring(n2)) +
          process(str.substring(0, n2));
  }
  return str;
}
```

("HAVE") = ("VE) + ("HA")
("VE")= ("E")+ ("V")
("E") = E
("V") = V
("HA") =("A") + ("H")
("A")= A
("H")= H

("H")= H
("A")= A
("HA")= A + H
("V")= V
("E")= E
("VE")= E + V
("HAVE")= E + V + A + H = EVAH

("FUN") = ("UN") + ("F")
("UN")= ("N") + ("U")
("N") = N
("U") = U
("F") = F

("F")= F
("U") = U
("N")= N
("UN") = N + U
("FUN")= N + U + F = NUF

What is the output from

```
System.out.println(process("HAVE") + " " + process("FUN"));
```

→ EVAH NUF

4. Write a recursive version of the following method.

```
// Returns the smallest among the first n elements of list
// Precondition: 1 <= n <= list.length
public int findMin(int[] list, int n)
```

Do not use any loops. ✓

5. Consider the following method:

```
public boolean isGood(String s)
{
  int n = s.length();
  return n < 2 || (s.charAt(0) == s.charAt(n-1) &&
                   isGood(s.substring(1, n-1)));
}
```

For which of the following strings will `isGood` return `true`?

■ (a) "" (empty string)
■ (b) "X"
■ (c) "XOX"
□ (d) "OXOX"
■ (e) "XOOX"
□ (f) "XXOX"
■ (g) "XXXX"
■ (h) "XOXOX"

every string that returns true is a palindrome, ones that are not return false

6. Suppose you have a method `printStars`, such that `printStars(n)` prints *n* stars on one line. For example, a statement

```
printStars(5);
```

displays the line

```
*****
```

(a) Using the `printStars` method, write a recursive method `printTriangle` so that `printTriangle(n)` prints a triangle with one star in the first row, two stars in the second row, and so on, up to `n` stars in the last row. For example, a statement

```
printTriangle(5);
```

should display

```
*
**
***
****
*****
```

Do not use any loops in your method.

(b) Modify your method `printTriangle` from Part (a) so that `printTriangle(5)` displays

```
*****
****
***
**
*
```

7. Consider

```
public void enigma(int n)
{
  for (int i = 0; i < n; i++)
    enigma(i);
  System.out.print(n);
}
```

(3) = (0) + (1) + (2) + 3
(2) = (0) + (1) + 2
(1) = (0) + 1
(0) = 0

(0) = 0
(1) = 0, 1
(2) = 0, 0, 1, 2
(3) = 0, 0, 1, 0, 0, 1, 2, 3

Does the call `enigma(3)` terminate? If so, what is the output?

0, 0, 1, 0, 0, 1, 2, 3

8. (a) Write a recursive method `sumDigits` that calculates and returns the sum of all the digits of a given non-negative integer. ✓

(b) Write a `boolean` method that tests whether a given number is evenly divisible by 3. A number is divisible by 3 if and only if the sum of its digits is divisible by 3. Use the `sumDigits` method from Part (a). Do not use any arithmetic operators in this method.

9. What is the output from the following method when called with $n = 3$? ✓

```
public void printX(int n)
{
  if (n <= 0)
    System.out.print(0);
  else
  {
    printX(n - 1);
    System.out.print(n);
    printX(n - 2);
  }
}
```

(3) = (2) + 3 + (1)
(2) = (1) + 2 + (0)
(1) = (0) + 1 + (-1)
(0) = 0
(-1) = 0

(-1) = 0
(0) = 0
(1) = 0 + 1 + 0 = 010
(2) = 010 + 2 + 0 = 01020
(3) = 01020 + 3 + 010 = 010203010

~~**10.** Rewrite the `gcf` method from Section 7.7 using recursion and not a single loop.~~

11. ▪ The following recursive method calculates 3^n:

```
// Precondition: n >= 0
public int power3(int n)
{
  if (n == 0)
    return 1;
  else
  {
    int p = power3(n/2);
    p *= p;

    if (n % 2 == 1)
      p *= 3;

    return p;
  }
}
```

(15) = (7) + 2
(7) = (3) + 2
(3) = (1) + 2
(1) = (0) + 2
8
(0) = 1
↳ return 0;

How many multiplications will be performed when the program calls `power3(15)`? 8

12. ▪ Consider the following recursive method:

```
public long someFun(int n)
{
  if (n <= 0)
    return 2;
  else
    return someFun(n-1) * someFun(n-1);
}
```

(5) = $\boxed{(4)}$ * $\boxed{(4)}$
(4) = $\boxed{(3)}$ * $\boxed{(3)}$ → calls method call of 3 twice
(3) = (2) * (2)
(2) = (1) * (1)
(1) = (0) * (0)
(0) = 2

(a) When the program calls `someFun(5)`, how many times will `someFun(3)` be called?

☆ (b) ▪ (MC) What does this method calculate when the input parameter *n* is a non-negative integer? ✓

A. n^2 1^2 B. 2^n 2^1 C. 2^{n+1} 2^2 D. 2^{2n+1} 2^3 [✓ E.] $2^{(2^n)}$ 2^2

13. What is the output from the following method when called with the argument $x = 2035$? ✓

```
public void display(int x)
{
  if (x >= 10)
  {
    display(x/10);
    System.out.print(x % 10);
  }
}
```

(2035) = (203) + 5
(203) = (20) + 3
(20) = (2) + 0
(2) = _

(20) = 0
(203) = 0+3
(2035) = 0+3+5 = $\boxed{035}$

14. ▪ The following recursive method returns the *n*-th Fibonacci number:

```
// Returns the n-th Fibonacci number.
// Precondition: n >= 1
```

```
    }
  }
```

14.■ The following recursive method returns the *n*-th Fibonacci number:

```
// Returns the n-th Fibonacci number.
// Precondition: n >= 1
public static long fibonacci(int n)
{
  if (n == 1 || n == 2)
    return 1;
  else
    return fibonacci(n - 1) + fibonacci(n - 2);
}
```

Rewrite it without recursion, using <u>one</u> loop.

15.■ The numbers $\binom{n}{0}, \binom{n}{1}, \binom{n}{2}, \ldots, \binom{n}{n}$ in the expansion

$$(x+y)^n = \binom{n}{0}x^n + \binom{n}{1}x^{n-1}y + \binom{n}{2}x^{n-2}y^2 + \ldots + \binom{n}{n-1}xy^{n-1} + \binom{n}{n}y^n$$

are called *binomial coefficients*. For example,

$(x+y)^2 = x^2 + 2xy + y^2$, so $\binom{2}{0}=1, \binom{2}{1}=2, \binom{2}{2}=1$.

$(x+y)^3 = x^3 + 3x^2y + 3xy^2 + y^3$, so $\binom{3}{0}=1, \binom{3}{1}=3, \binom{3}{2}=3, \binom{3}{3}=1$.

$(x+y)^4 = x^4 + 4x^3y + 6x^2y^2 + 4xy^3 + y^4$, so

$\binom{4}{0}=1, \binom{4}{1}=4, \binom{4}{2}=6, \binom{4}{3}=4, \binom{4}{4}=1$.

$\binom{n}{k}$ is pronounced "n-choose-k" and sometimes written as *C*(*n*, *k*).

Binomial coefficients have the following properties: for any $n \geq 0$,

$\binom{n}{0} = \binom{n}{n} = 1$, and for any integers *n* and *k*, such that $0 < k < n$,

$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$.

Complete the <u>recursive</u> method `binomialCoeff` below, which computes a specified binomial coefficient.

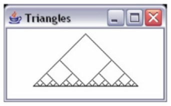
```
// Returns the value of the binomial coefficient C(n, k)
// Precondition: 0 <= k <= n
public int binomialCoeff(int n, int k)
{
  ...
}
```

16. Suppose the `File` class from Section 13.4 has a method `getSize()`, which returns the size of the file in bytes. Let's say that the size of a folder includes 512 bytes for the description of the folder itself, plus 128 bytes for the directory entry for each item (file or subfolder) in the folder, plus the sizes in bytes of all the items in the folder. Write the `getSize` method for a `Folder` that returns the size in bytes of the folder and all its subfolders and all the files in all the subfolders. ✓

~~**17.**■ Suppose we have added the `String getName()` method to the `FileItem` interface from Section 13.4 and implemented this method in the `File` and `Folder` classes. Write `boolean` methods `contains(String name)` for the `File` and `Folder` classes. For a `File`, `contains` should always return `false`. For a `Folder`, the method should return `true` if this folder or any of its subfolders contain a file or folder with the given name; otherwise it should return `false`.~~

18. Consider a sequence $x_1 = 1$, $x_2 = 1 + \frac{1}{1}$, $x_3 = 1 + \frac{1}{1+\frac{1}{1}}$, ... In this sequence $x_{n+1} = 1 + \frac{1}{x_n}$ (for $n \geq 1$). In Chapter 7 Question 17 you wrote an iterative version of a method that computes x_n. Now write a <u>recursive</u> version of this method.

19.■ The program *Fractal* (`JM\Ch13\Exercises\Fractal.java`) displays a picture made of nested right isosceles triangles, as shown below.



The half-length of the base of the smallest triangle is 4. In the picture, the half-length of the base of the largest triangle is 64, and the coordinates of the midpoint of the base are (100, 100). Fill in the blanks in the recursive method `drawTriangles`. ⊰ Hint: a statement

```
        g.drawLine(x1, y1, x2, y2);
```

draws a line from point `(x1, y1)` to point `(x2, y2)` in the graphics context `g`. ⊱