CS 106A Section 2 Handout (Week 3)

Console Programs

1. **Fibonacci Sequence**. Write a `ConsoleProgram` that displays all the numbers in the Fibonacci Sequence that are less than 10,000, starting with 0. The Italian mathematician Leonardo Fibonacci devised the Fibonacci sequence as a way to model the growth of a population of rabbits. The first two terms in the sequence are 0 and 1, and every subsequent term is a sum of the previous two terms. You should print the output as shown to the right.

   The Fibonacci sequence has numerous applications in computer science and shows up in surprising places. It's used to compute logarithms, index and retrieve data, and as a building block in some route-planning algorithms.

```
Fibonacci sequence:
0
1
1
2
3
5
8
13
…
```

2. **If/Else Mystery.** For each call to the following method, indicate what output is produced.

```
public void mystery(int n) {
    print(n + " ");
    if (n > 0) {
        n = n - 5;
    }
    if (n < 0) {
        n = n + 7;
    } else {
        n = n * 2;
    }
    println(n);
}
```

Call	**Output**
`mystery(8);`	____________________
`mystery(-3);`	____________________
`mystery(1);`	____________________
`mystery(0);`	____________________

3. **AsciiFigure**. Write a `ConsoleProgram` that draws a figure of the following form, using for loops.

```
////////////////\\\\\\\\\\\\\\\\
////////////********\\\\\\\\\\\\
////////****************\\\\\\\\
////************************\\\\
********************************
```

   Then, modify your program so that it can create a similar figure of any size. For instance, the diagram above has a size of 5. The figure on the left has a size of 3, and the figure on the right has a size of 7.

```
////////\\\\\\\\
////********\\\\
****************
```

```
////////////////////////\\\\\\\\\\\\\\\\\\\\\\\\
////////////////////********\\\\\\\\\\\\\\\\\\\\
////////////////****************\\\\\\\\\\\\\\\\
////////////************************\\\\\\\\\\\\
////////********************************\\\\\\\\
////****************************************\\\\
************************************************
```

4. **ShowTwos.** Write a method `showTwos` that shows the factors of 2 in an integer. For example:

```
Call              Output
showTwos(7);      7 = 7
showTwos(4);      4 = 2 * 2 * 1
showTwos(18);     18 = 2 * 9
showTwos(120);    120 = 2 * 2 * 2 * 15
```

   The idea is to express the number as a product of factors of 2 and an odd number. The number 120 has 3 factors of 2 multiplied by the odd number 15. For odd numbers (e.g. 7), there are no factors of 2, so you just show the number itself. Assume that your method is passed a number greater than 0.

   Then write a `run` method that calls `showTwos` with 10 randomly chosen integers (between 0 and 500, inclusive).

CS 106A Section 2 Handout (Week 3)

Parameters and Return

Trace through the execution of the programs below and show what output is produced when they run.

5. Trace with Parameters

```java
import acm.program.*;

public class ParameterMystery1
            extends ConsoleProgram {
    public void run() {
        int a = 4;
        int b = 7;
        int c = -2;

        m(a, b, c);
        m(c, 3, a);
        m(a + b, b + c, c + a);
    }

    public void m(int c, int a, int b) {
        println(b + " + " + c + " = " + a);
    }
}
```

6. Trace with Parameters and Returns

```java
import acm.program.*;

public class ParamAndReturn
            extends ConsoleProgram {
    public void run() {
        int a = 137;
        int b = 42;

        println("a = " + a);
        foo(b);
        println("a = " + a);
        println("b = " + b);

        a = bar(b, a + b);
        println("a = " + a);
        a = bar(a, b);
        println("a = " + a);
    }

    public void foo(int a) {
        println("a = " + a);
        a = 160;
    }

    public int bar(int c, int b) {
        int d = b - c;
        println("d = " + d);
        return d % 10;
    }
}
```

7. Days in Month. Write a method named `daysInMonth` that accepts a month (an integer between 1 and 12) as a parameter and returns the number of days in that month. For example, `daysInMonth(9)` returns `30` because September has 30 days. Ignore leap years; assume that February always has 28 days.

Month	1 Jan	2 Feb	3 Mar	4 Apr	5 May	6 Jun	7 Jul	8 Aug	9 Sep	10 Oct	11 Nov	12 Dec
Days	31	28	31	30	31	30	31	31	30	31	30	31

Optionally, also accept a year parameter and return 29 days for February in leap years. Leap years are divisible by 4 but not by 100 unless by 400, so 2016 and 1868 are leap years, while 1933 and 2018 are not.

8. Piglet. Write the code for a simple 1-player dice game called "Piglet" (based on the game "Pig"). The player's goal is to accumulate as many points as possible without rolling a 1. Each turn, the player rolls the die; if they roll a 1, the game ends and they get a score of 0. Otherwise, they add this number to their running total score. They then choose whether to roll again, or end the game with their current point total. Two sample games are shown to the right.

```
Welcome to Piglet!
You rolled a 5!
Roll again? yes
You rolled a 4!
Roll again? yes
You rolled a 1!
You got 0 points.
```

```
Welcome to Piglet!
You rolled a 6!
Roll again? yes
You rolled a 2!
Roll again? yes
You rolled a 2!
Roll again? no
You got 10 points.
```

Tip: use the `readBoolean` method to prompt the user with a yes/no question.