Problem #1: Write a method that calculates n! for any integer n. In other words, your method should pass in an integer n, and it should return an integer (that is equal to n!). If you don’t get this one right, then you have not been in class (hint, hint).

Problem #2: Write a method that calculates \(x^n\) for any integer n and double x. Use a “for” loop to multiply x the correct number of times. Now use your method (in the “main”) for the numbers x = 4 and n = 3.

Problem #3: Write a method that takes an integer n as a parameter and returns an integer. The integer that it returns should be 2n - 1. (By the way, notice that for n = 1, 2, 3, 4, 5… this will give back the numbers 1, 3, 5, 7, 9… In other words, it creates the odd numbers. This will be handy in the next problem.)

Problem #4: Calculators don’t have magical methods of determining sin and cos. Instead, they are programmed to approximate sin and cos from “infinite series”. So let’s do the same thing! Write code that calculates \(\sin(x)\) by using its series.

\[
\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \frac{x^{11}}{11!}
\]

The true series uses an infinite number of terms, but your code should use only 6 terms as shown. Your code should also ask the user for a value of “x”, and then it should print the answer.

Hints: Use your methods from problems #1, #2, and #3! Also, watch out for that pesky integer division. You may need to cast your factorial values to doubles. (Note: with only 6 terms, this series will give bad answers for large x. But if you keep x between roughly 1 and -1 everything will work well. You may use more
terms for better accuracy, but then you can’t use “int” variables because the numbers like $x^{13}$ and $13!$ get too big.

**Extra Credit (2 points):** Solve problem #4 using 10 terms. Why is this a little bit tricky? Because the factorial gets huge – so huge that after 6 terms it is bigger than an “int” is allowed to get. In Java, an “int” can only be between $-2^{31}$ and $2^{31}$. 