Please complete the textbook sections through 2.2 before beginning this assignment.

CS 278
Lab 4: Disproof by Counterexample

All programs you submit in this class must follow the Documentation and Style Guidelines. This document can be found in the Canvas Modules.

All programs you submit in this class must compile with the Oracle Java compiler on the Linux machines in SH 118 or SH 118B.

Introduction

Beginning in the 1970's, many proofs have been generated by computer. In 1976, the Four Color Theorem was the first major theorem to be verified using a computer program.

Finding a specific counterexample to a statement by hand can be a time-consuming task. However, for many theorems, finding a counterexample using a computer program is quite easy. If you can express the statement mathematically, you can write a computer program that tests specific values from the domain, one-by-one until a counterexample is found (or until it reaches some upper limit and you decide that there is no counterexample).

The Big Picture

In this assignment you will write a computer program to disprove a statement by finding a counterexample.

The statement is the following:

Every odd number x, such that x >= 3, can be written as the sum of a prime number and twice a square.

Domain: The domain is all odd integers greater than or equal to 3.

Predicate: P(x) means that x = a prime + 2(a perfect square)

Expressing the statement in logic: $\forall x\, P(x)$

Remember that to disprove a universal statement, we only have to find one value for x that makes the predicate false.

The predicate is indeed true for some values of x. Here are some examples:

x = 3	3 = 3 + 2 (0)	3 is a prime and 0 is a perfect square
x = 5	5 = 3 + 2 (1)	3 is a prime and 1 is a perfect square
x = 7	7 = 5 + 2 (1)	5 is a prime and 1 is a perfect square

9, 11, 13... and quite a few more odd numbers after that fit the requirement. It fools us into thinking that this is a pattern that continues forever. But it doesn't.

There are some counterexamples. The smallest counterexample is between 1,000 and 10,000.

Your task is to find the smallest counterexample; specifically, the smallest odd number greater than or equal to 3 that cannot be written as the sum of a prime number and twice a square.

Note: You do not need to copy any methods from PA 1, 2, or 3.

Now for the Details

This is not just a suggestion. Follow the steps given below.

1. Start a new Java program called PA4.java.

   Your program must follow the course Documentation and Style requirements.

2. Write a method named buildSquareArray that accepts an array of integers as its parameter.

   Fill the array with the squares of integers.

   Specifically...

   store 0^2 in array[0]

   store 1^2 in array[1]

   and so on

   All methods will be public and static.

3. In the main, test the buildSquareArray method with these steps:

   a. Create and instantiate an array of 100 integers. Name this array perfectSquares.

      Note: 100 elements is enough because there is a counterexample n, where 1000 < n < 10000.

   b. Call the buildSquareArray method.

   c. To verify that buildSquareArray worked correctly. Print the first 10 elements of the array, all on one line, with one space between.

      Expected Output: 0 1 4 9 16 25 36 49 64 81

   Once you are sure that the method works correctly, delete the code that prints the array.

4. Write a method named isPrime that returns a boolean value (true if the integer parameter is prime, false if it is not).

5. In the main method, test the isPrime method with the following values:

Parameter Value	Expected Return
-3	false
1	false
2	true
19	true
49	false
91	false

Once you are sure that the method works correctly, delete the testing statements.

6. Now, rethink the theorem we are trying to <u>disprove</u>.

For every odd number x greater than or equal to 3, x = a prime + 2(a perfect square).

Let p = "some prime number" and s = "a perfect square".

Then we can write an equation for x, x = p + 2s.

Now, we solve for p.

p = x - 2s

If we have some integer value for x, we can calculate p using this equation. And... if p is indeed a prime number, then the theorem holds.

But... if p is not a prime number, we have a counterexample.

7. What you have in your program so far is

- a method that can determine whether a number is prime or not
- a method that builds an array that contains perfect squares

In the main method, take the following approach to solving the problem:

a) Note: In the explanation above, the odd number is called x. Other variables given in the problem are p and s. Variable names should match what is given in the problem.

b) Create a flag variable named theoremHolds and set it to true.

c) Use a while loop. The loop should repeat as long as x is less than or equal to 9999 (odd numbers only) and the theorem holds {

- Assume the theorem does not hold (set the flag variable to false).
- Use a for loop. for every square (elements come from the array perfectSquares) until the theorem holds or you reach a square that is greater than 1/2 x {

    calculate p

    if p is prime, the theorem holds for this x, so set the flag variable to true

  } // end for loop

- If the theorem did not hold, then x is a counterexample. The outer loop should stop.
  (remember... we only need one counterexample to disprove the theorem)

} // end while loop

Using a break statement is not allowed.

d) If a counterexample was found...print a message and the value of x. Otherwise, print a message saying no counterexample was found.

Submit PA4.java on Canvas.