CS 136 Fall 2008
Mark Giesbrecht

Assignment 3
Due Tuesday, October 14, 2008 at 4pm.

All questions are to be implemented in the C language

1. Write a function `isPerfectPower` which consumes a positive `int x`, and returns the smallest `int a` such that there exists a positive integer k with $x = a^k$. If no such `a` and k exist, then `isPerfectPower` returns 0. For example `isPerfectPower(16)` return 2, and `isPerfectPower(27)` returns 3, while `isPerfectPower(15)` returns 0.

   As discussed in class, put your `isPerfectPower` function (and any helper functions you might need) in a file called `isPerfectPower.c` (with no `main()` function). Put a prototype for the function `isPerfectPower` in the file `isPerfectPower.h`. Put a `main()` function in a file called `isPerfectPower-driver.c`, and include appropriate test cases for the `isPerfectPower` function using `assert`.

2. Pell's equation has been studied since at least 400 BCE. Given a number n (which is not a perfect square), it seeks to find positive integers x and y such that

   $$x^2 - ny^2 = 1$$

   For example, if $n = 7$ then the (x, y) pairs

   $$(8, 3); (127, 48); (2024, 765); (32257, 12192); (514088, 194307); (8193151; 3096720); ...$$

   are solutions to Pell's equation. Lagrange proved that an infinite number of such x and y always exist in 1766!

   In this question you are to write a function `pell` which consumes an argument `int n`. It produces the smallest positive `int x` such that there exists a `int y` with $\texttt{x}^2 - \texttt{ny}^2 = 1$. For example, with $\texttt{n} = 7$ as above, you would return 8.

   Put your `pell` function (and any helper functions you might need) in a file called `pell.c` (with no `main()` function). Put a prototype for the function `pell` in the file `pell.h`. Put a `main()` function in a file called `pell-driver.c`, and include appropriate test cases for the `pell` function using `assert`.

3. A famous mathematical problem, known as the Collatz conjecture, or $3n+1$ problem, considers the following operation. Given any positive integer n, if n is even then divide it by two. If n is odd, triple it and add one. Repeat this operation over and over again. The conjecture states that eventually the sequence will reach the number one.

   For example, starting with $n = 6$, one gets the sequence 6, 3, 10, 5, 16, 8, 4, 2, 1. The *stopping time* is the number of times we need to apply the rule before one is reached. In the above example the stopping time is 8.

   (a) Write a function which `collatzStop` which consumes a single parameter `int x` and returns an `int` which is the stopping time for the $3n + 1$ problem on input `x`.

(b) Write a second function `biggestCollatz` which consumes two `int` parameters `lo` and `hi`, and returns the number x with $\texttt{lo} \leq x \leq \texttt{hi}$ with largest Collatz stopping time. If multiple numbers in the range have the same Collatz stopping time, return the smallest such number.

Put your `collatzStop` and `biggestCollatz` functions (and any helper functions you might need) in a file called `collatz.c` (with no `main()` function). Put prototypes for the functions `collatzStop` and `biggestCollatz` in the file `collatz.h`. Put a `main()` function in a file called `collatz-driver.c`, and include appropriate test cases for the `collatzStop` and `biggestCollatz` functions using `assert`.