CS 70 Discrete Mathematics and Probability Theory
Summer 2011 Kamil PA 1

Due Friday, July 1, 11:59pm

You must complete this programming assignment on your own. Remember that you have three slip days that you can allocate between the three programming assignments. Note that there is a short grace period (on the order of minutes, not hours) after the above deadline to account for clock discrepancies, last minute submission hiccups, etc.

Overview

As we have seen in class, there is a strong connection between induction and recursion. Both involve breaking down a problem or proof in terms of smaller instances. Both have base cases and an inductive or recursive step. In this programming assignment, we further explore this connection by writing a recursive program to generate direct proofs for a particular claim. The structure of this program will follow the general structure of an inductive proof of that claim.

Consider the following theorem:

> Every natural number $n > 1$ can be written as a product of primes.

An inductive proof of this theorem is provided in Note 3 (page 25). Given a particular $n > 1$, your job is to generate a direct proof that n can be written as a product of primes. For example, if given $n = 5$, your program should generate output similar to the following:

```
5 is prime, so it is a product of a single prime.
```

On the other hand, given $n = 60$, your program should output:

```
In order to prove that 60 is a product of primes
  we prove that 6 and 10 are products of primes.
In order to prove that 6 is a product of primes
  we prove that 2 and 3 are products of primes.
2 is prime, so it is a product of a single prime.
3 is prime, so it is a product of a single prime.
Having proven that 2 and 3 are products of primes,
  we conclude that 6 = 2 * 3
  is a product of primes.
In order to prove that 10 is a product of primes
  we prove that 2 and 5 are products of primes.
2 is prime, so it is a product of a single prime.
5 is prime, so it is a product of a single prime.
```

```
Having proven that 2 and 5 are products of primes,
  we conclude that 10 = 2 * 5
  is a product of primes.
Having proven that 6 and 10 are products of primes,
  we conclude that 60 = 6 * 10
  is a product of primes.
```

The generated proof breaks 60 into 6 and 10, proves that 6 and 10 can each be written as a product of primes, and concludes as a result that $60 = 6 \times 10$ can also be written as a product of primes.

For this assignment, you must write the direct proof generating program as described above. **Your program must use recursion to generate the proof in order to receive any credit.** You may use any of the following programming languages: Scheme, Java, Python, C, or C++. (Note that though you may use any of these languages, we can only provide you skeletons in a subset of the languages. It may be beneficial to look over a skeleton for another language if one is not available in the language of your choice.) You have access to `stk`, `javac` and `java`, `python`, `gcc`, and `g++` from your class accounts. For the latter four languages, your program should take a single integer $n > 1$ as a command line argument and then print out a direct proof that n can be written as a product of primes. Skeletons for Scheme, Java, and Python with some useful auxiliary functions are provided at `http://www.cs.berkeley.edu/~kamil/cs70/pa/pa1/`.

Your program need not perform any error checking (e.g. that the given input is a positive integer greater than 1). You may subdivide a given integer n into any two factors you like (e.g. $60 = 6 \times 10$, $60 = 5 \times 12$, $60 = 4 \times 15$, $60 = 3 \times 20$, or $60 = 2 \times 30$). Stylistically, it is preferable that you use the factors closest to $\sqrt{n}$, though that is not required for this assignment. It is perfectly acceptable to repeat the proof for a particular factor m of n if n is divisible by a power of m, as demonstrated by the multiple occurrences of $m = 2$ in the sample output for $n = 60$.

Special Notes for Scheme

If you use Scheme, you should instead define a function called `prove` that takes in a single argument. When called with an integer $n > 1$, the function should display a direct proof that n can be written as a product of primes.

Some built-in Scheme functions that may prove helpful are `string-append`, `number->string`, and `display`.

Submission Instructions

You must submit your assignment electronically with the following command from your class account:

```
submit pa1
```

Two files are required in the directory from which you run the above command. A `README` file described below and a ZIP archive called `pa1.zip`. This archive should include all files necessary to compile and run your program. You can construct this archive using the following command:

```
zip pa1.zip <file1> ...  <fileN>
```

If you are adding directories, you will need to pass the `-r` flag to `zip`. As a sanity check, run

```
unzip -l pa1.zip
```

afterwards to make sure that all intended files are included in the archive. **We will not be able to grade incomplete submissions, so make sure to double check that all files are included.**

The README File

In your `README` file, include explicit instructions on how to build and run your code. In addition, answer the following questions:

(a) Does your program use the same base case as the inductive proof in the notes? If not, explain what you changed and why.

(b) Did this assignment help you in understanding induction?

(c) How long did this assignment take you to complete?