Lab 4 - Numbers

Due: end of class

OVERVIEW

As you know, a numeric type in C is only an approximation to the mathematical entity due to bit-size limitations. The goal of this lab is to deal with real problems caused by such differences. Along the way, you will also get more practice using loops.

Recall that n! (read n factorial) is defined to be (1)(2)(3)(4)...(n-1)(n). Many useful computer science problems require the computation of C(n,k) (read n choose k) which is defined as: n! / (k! * (n-k)!), for n>=0 and 0<=k<=n. For this lab, you will compute C(n,k) in several ways.

Two useful properties about C(n,k) are:

- (P1) C(n,n-1) = n
- (P2) C(n,k) = C(n,n-k)

Several tasks in this assignment ask you for an explanation. You don't need a detailed explanation (1 sentence each suffices), as long as you are clear. You may write your explanations in comments. Also, you may assume that all inputs are legal for this lab (so, no error checking needed).

TASKS

1. Implement a program that inputs n and k and then computes C(n,k) directly by first computing the 3 factorials: n!, k!, and (n-k)!. You probably want to do this by writing one loop each, mostly cut/paste of the first loop. You should represent variables as ints. What is the smallest value of n for which C(n,n-1) returns an incorrect result? Explain why the results are incorrect (you don't need to figure out the actual relevant bit patterns).
2. C(n,k) can be rewritten as n(n-1)(n-2)...(n-k+1)/k!. Implement a program that computes C(n,k) based on this formula (Hint: the numerator is obviously an iteration, but what is the iteration variable?). Repeat the questions from the previous part. This program is obviously better than 1 since it works on more input values, but how does it compare to 1 efficiency-wise? An appropriate measure of efficiency here would be the number of multiplies.
3. Use property (P2) to make your solution to 3 more efficient (in some cases).
4. The formula for C(n,k) can be expressed as the product (1+(n-k)/1) (1+(n-k)/2) (1+(n-k)/3) ... (1+(n-k)/k). Write a program to implement this. Note that the terms in the product may not be integers. You should be able to handle much larger values of n and k now.
   Warning: a naive implementation that ignores the type casting needed to convert between ints and intermediate data types will result in incorrect results - you can compare the results from previous tasks and this.

There are better algorithms that require only integer arithmetic, though we do not have enough C knowledge yet to implement these. If you wish to learn more, you can look up dynamic programming algorithms (and arrays to implement them).

HAND IN

Your 136 instructor will tell you what to hand in and how.

GENERAL COMMENTS FOR ALL PROGRAMS THIS SEMESTER

You should have the following header on all programs:

```
/*
  Author: <name>
  Course: {135,136}
  Instructor: <name>
  Assignment: <title, e.g., "Lab 1">

  This program does ...
*/
```

GRADING

All 135 and 136 programs this semester will be graded on:

- Correctness: Does your program work?
- Testing: Have you generated sufficient and good test data to give reasonable confidence that your program works?
- Structure: Have you structured your code to follow proper software engineering guidelines? This includes readability and maintainability.
- Documentation: How well documented is your code? Good documentation does not repeat the code in English, but explains the point of each code block, highlighting any design decisions and/or tricky implementation details.