PIC 10B SPRING 2013 HOMEWORK 5

Mathematical Background

An integer partition of length n is a set of positive integers that sum up to n. For example, the integer partitions of 5 are $\{5\}, \{4,1\}, \{3,2\}, \{3,1,1\}, \{2,2,1\}, \{2,1,1,1\}, \{1,1,1,1,1\}$; the order does not matter, so $\{2,2,1\}$ is the same as $\{1,2,2\}$ and $\{2,1,2\}$, and so we write them in decreasing order. The listing of all integer partitions for a given n is just a little too complicated for our course, so instead we will focus on a much simpler recursive structure that involves counting them.

Denote by $p(k,n)$ the number of integer partitions of n with each element (also called *part*) at most k. So for example,

- $p(1,5) = 1$ because there is exactly one partition of 5 whose parts are *all* at most 1, namely, $\{1,1,1,1,1\}$.
- $p(2,5) = 3$ because there are exactly three partitions of 5 whose parts are at most 2, namely, $\{2,2,1\}$, $\{2,1,1,1\}$, and $\{1,1,1,1,1\}$.
- $p(3,5) = 5$ because there are exactly five partitions of 5 whose parts are at most 3, namely, $\{3,2\}$, $\{3,1,1\}$, $\{2,2,1\}$, $\{2,1,1,1\}$, $\{1,1,1,1,1\}$.
- $p(4,5) = 6$ because there are exactly six partition of 5 whose parts are at most 4, namely, $\{4,1\}$, $\{3,2\}$, $\{3,1,1\}$, $\{2,2,1\}$, $\{2,1,1,1\}$, $\{1,1,1,1,1\}$.
- $p(5,5) = 7$ because there are exactly seven partitions of 5 total (listed in the above paragraph).

We will address in this assignment: how do you compute $p(k,n)$ for arbitrary values of k and n? This is where recursion comes into play. There is a simple argument for why the following recursive formula is true:

$$p(k,n) = p(k,n-k) + p(k-1,n), \quad 1 \leq k \leq n.$$

There is no need to understand WHY the formula is correct in order to do the assignment[1].

Now we need to establish what happens when $n \leq 0$ or $k \leq 0$. Again, it is not necessary to understand WHY, it is simply sufficient to know the following formulas, which are *base cases*:

- $p(k,n) = 0$ for $k < 0$ OR $n < 0$. I.e., any negative input is automatically 0.
- $p(0,n) = 0$ when $n > 0$.
- $p(k,0) = 1$ when $k \geq 0$.

There is finally one other case, which is

- $p(k,n) = p(n,n)$ when $k > n$.

Date: May 1, 2013.

[1]But if you are curious, here is a simply worded argument (if it helps plug in $n = 5$ and $k = 2$): think about all the partitions of n with all parts at most k. Each of those partitions either has a k in the set or does NOT have a k. If they have a k, subtract it from the total and then count the number of partitions of $n-k$ with parts at most k (because we could have multiple copies of k). If the set does NOT have k, then the largest part is at most $k-1$ and so we wish to count the number of partitions of n into parts at most of size $k-1$.

Assignment

Create a function that computes the *number* of integer partitions of n with each element at most k. The program should allow the user to input a positive integer m (you may assume the input will indeed be a positive integer, no need to error check), print out $p(k, m)$ for $k = 1, 2, \ldots, m$, and then create a file named *pknTable.txt* that creates a table of all values for $1 \leq k \leq m$ and $1 \leq n \leq m$.

Place your code in a source file labeled *hw5.cpp*. ***If your file is not named this exactly, your homework will not be collected***. As with all programs in this course, your code should contain useful comments. In particular, your name, the date, and a brief description of what the program does should appear at the top of your source file.

What to Turn in

Place in your Submit folder the source file *hw5.cpp* with exactly this name (all lowercase, no spaces). The files will be automatically collected on Friday 5/3/13 at 5:00pm.

Grading		
Recursion	Correctly implement a recursive solution	5 points
Output File	Correctly creates a text file with the appropriate values	5 points
Solution	Code is efficient but easy to follow	5 points
Style	Variable names, comments, indentation	5 points
	TOTAL	20 points

Note on grading: There is an automatic 5 point penalty for any homework that does not compile.

A *sample* of output is below. NOTE! The values input are just an example. When grading the homework the grader will select a value of m.

```
Welcome! Please input a number m: 5

p(1,5) = 1
p(2,5) = 3
p(3,5) = 5
p(4,5) = 6
p(5,5) = 7

Creating file ... Done.
Press any key to continue...
```

Your text file should look like the following:

1	1	1	1	1
1	2	2	3	3
1	2	3	4	5
1	2	3	5	6
1	2	3	5	7

In other words, the entries in the matrix are

$$p(1,1)\ p(1,2)\ p(1,3)\ ...$$
$$p(2,1)\ p(2,2)\ p(2,3)\ ...$$
$$p(3,1)\ p(3,2)\ p(3,3)\ ...$$
$$...$$

NOTE: Do not worry about the digits not lining up for larger values of m. Just make sure the formatting is correct and that there is exactly one space between each integer value, and a return carriage between lines.

Due to the large amount of internal stack memory required for this recursion, you may assume that the input value of m to your program will be less than 30. Larger values of m can cause the program to crash or exit unexpectedly, due to the stack running out of memory with all of the recursive calls required. (You should play around with large values of m just to see what the limits are.)