Project 2: Dynamic programming

COT 4400, Spring 2016

Due April 1, 2016

1 Overview

This project requires you to theoretically solve a dynamic programming problem and write a program that implements your solution.

You are only allowed to consult the class slides, the textbook, the TAs, and the professor. In particular, you are not allowed to use the Internet. This is a group project. The only people you can work with on this project are your group members. This policy is strictly enforced.

In addition to the group submission, you will also evaluate your teammates' cooperation and contribution. You will submit these evaluations to another assignment on Canvas, labelled "Project 2 (individual)."

2 Problem Description

For this project, you will solve the *(b, n, k)-basket* problem. This problem is a generic combinatorial problem in which you have some number of baskets (n), each of which can hold up to k balls, and you need to determine how many ways that a given number of balls (b) can be distributed into the baskets.

Of course, the solution to this problem could be applied in many different contexts. As an example, you might ask how many chocolate samples we could put together out of n types of chocolates so that no sample contains more than two of the same type of chocolate. In this example, the "balls" are the number of pieces in the sample, while the "baskets" are the types of chocolates (with a limit of 2 chocolates per type). Given an assignment of balls to baskets, we could create a chocolate sample by adding a piece of one type of chocolate to the sample for each ball in the corresponding basket.

Your solution to this problem *must use dynamic programming*. Solutions based purely on combinatorics will not receive credit.

3 Modeling the problem

Before you can write a program to solve this problem, you must first write a report describing how you will solve this problem. This report should address the following:

1. Describe in English how you can break down the larger problem into one or more smaller problem(s). This description should include how the solution to the larger problem is constructed from the subproblems.

2. What recurrence can you use to model the problem using dynamic programming?
3. What are the base cases of this recurrence?
4. Describe a pseudocode algorithm that uses memoization to solve the problem.
5. Describe a pseudocode algorithm that solves the problem iteratively (using dynamic programming). Your algorithm should be optimal in terms of its time and space complexity.
6. Analyze the complexity of your iterative algorithm in terms of b, n, and k.
7. What is the space complexity of your iterative algorithm?

For full credit, your pseudocode must be clear enough that any competent programmer will understand how your algorithm works and could implement your algorithm in their preferred programming language.

4 Coding your solutions

In addition to the report, you should implement the *memoized* version of your algorithm so that it can solve the (b, n, k)-basket problem. Your code may be in C++ or Java, but it must compile and run on the C4 Linux Lab machines. Your code may be split into any number of files.

4.1 Input format

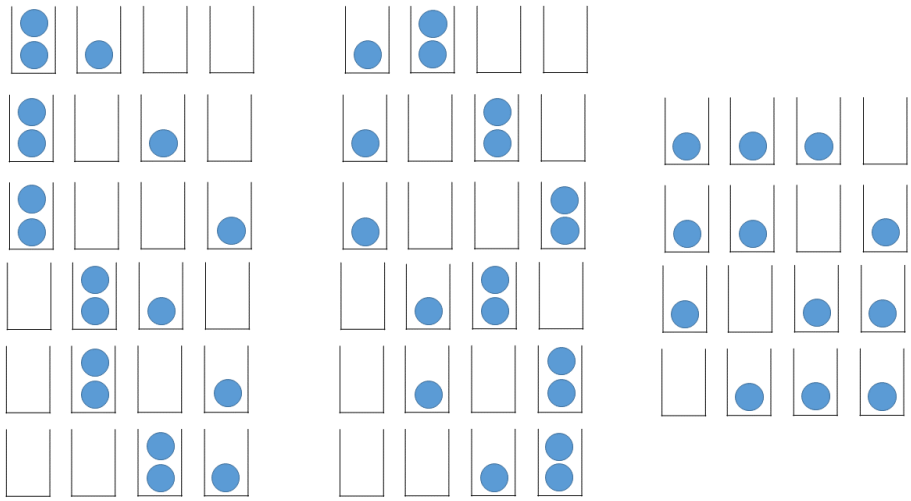
Your program should read its input from the file `input.txt`, in the following format. This file may contains multiple instances. It begins with a single positive integer on a line by itself indicating the number of problem instances in the file. The remaining lines of the file each contain three integers separated by a space, representing the values of b, n, and k (in that order) for one problem instance.

4.2 Output format

Your program should write its output to the file `output.txt`. You should write one line for each instance in the input file, and each line should contain the solution to that instance.

4.3 Example

When $b = 3$, $n = 4$, and $k = 2$, there are 16 ways to distribute the three balls among the four baskets, as shown below:



When checking your solution, you may use the fact that the solution to the (b, n, k)-basket problem is $\binom{n}{b}$ when $k = 1$ and $\binom{n+b-1}{b}$ when $k = b$. Moreover, the solution strictly increases as k moves from 1 to b. Why these statements are true is left as an exercise to the reader.

5 Submission

Your submission for this project will be in two parts, the group submission and your individual peer evaluations.

The submission for your group should be a zip archive containing 1) your report (described in Section 3) as a PDF document, 2) your code (described in Section 4), and 3) a README file describing how to compile and run your code to Canvas. If your code requires more than a simple command to compile and run then you must also provide a Makefile and/or shell script. A simple command might be something like:

```
g++ *.cpp -o basket
```

Be aware that your project report and code will be checked for plagiarism.

6 Teamwork evaluation

The second part of your submission is a text file that includes 1) the names of all of your teammates (including yourself), 2) the team member responsibilities, 3) whether or not your teammates were cooperative, 4) a numeric rating indicating the proportional amount of effort each of you put into the project, and 5) other issues we should be aware of when evaluating your and your teammates' relative contribution. The numeric ratings must be integers that sum to 30.

7 Grading

Report	**40 points**
Dynamic programming model	20
Memoized pseudocode	5
Iterative pseudocode	5
Iterative complexity	10
Code	**30 points**
README file	5
Compiles and is correct	20
Good coding style	5
Teamwork	**30 points**
Follows the given format	5
Participation	25

Note that if your algorithm is inefficient, you may lose points for both your pseudocode and your submission. Also, in extreme cases, the teamwork portion of your grade may become negative. In particular, if you do not contribute to your group's solution at all, you can expect to receive a 0 overall.