For this lab, you will write a program that implements a solution to the **0/1 Knapsack** problem. An explanation of the problem and the pseudocode for the solution is available on the class web page. In this lab, you will implement all three versions that are given in the pseudocode: One that uses recursion; one that uses dynamic programming; and one that uses memoization. You may write three separate programs or one program that implements all three algorithms. If you write one program, please offer the user a choice of which algorithm to use. The user interface may be graphical, or it may be a simple menu system. We are primarily interested in the algorithms, so the user interface just needs to be convenient to use and functional. The input to the program will be an integer, which represents the maximum weight the knapsack can hold.

Note that the recursive solution only calculates the maximum profit. It does not tell you how many of which objects constitute the maximum profit. To be able to print the objects chosen as well as the maximum profit requires the use of a second array. The pseudocode for the dynamic programming solution indicates how to record and how to print the objects used for the maximum profit.

**One difference, however, between this lab and the pseudocode on the class web page is the weight and profit of the objects to be used.** For this lab use the following.

| | **Object 1** | **Object 2** | **Object 3** | **Object 4** |
|---|---|---|---|---|
| **Profit** | 9 | 7 | 5 | 1 |
| **Weight** | 8 | 7 | 6 | 2 |

For example, if the knapsack capacity was 17, the maximum profit is obtained by taking two of object 1 for a total profit of 18.

**To Turn In**

Turn in a copy of your source code by e-mail in an archive file named XXXL3.zip, where XXX represents your initials. This file can be either a **zip** file or a **jar** file. **Be sure that your program files contain a header with your name on it and the name of your archive file**. In addition, you need to turn in a hard copy of the program.

This lab is worth 55 points.

5 points: The archive file contains only source files, is e-mailed on time, and your name is in a comment in the header.

5 points: A hard copy of your source files is handed in, and the name of the archive file is contained in a comment in the header.

5 points: The source files have appropriate documentation (comments) and use a consistent and readable programming style.

5 points: The source files compile and run under JDK 1.5 or higher.

5 points: A **recursive** implementation that correctly determines the maximum profit for the user-entered weight.

10 points: A **dynamic programming** solution that correctly solves the problem.
5 points: The dynamic programming solution outputs the objects used for the maximum profit.

10 points: A **memoized** solution that correctly solves the problem.
5 points: The memoized solution outputs the objects used for the maximum profit.