Thievery

Much to the chagrin of your teacher, you didn't work hard in CS 3. This caused you to fail all your computer science college courses, which lead to you flunking out of college entirely. Mechanization is eliminating all the fast food jobs, so you're left with one option: *burglary*.

Houses no longer keep money in safes, having been previously robbed (by you). They do contain other valuable things though! When robbing your first house, you come to a startling realization. No, not that you've made some bad choices and need to make some positive changes in your life. You realize that there are a lot of items in a house to steal, but you can only carry so much at once, and a wise burglar doesn't stick around weighing the relative merits of stealing a television vs. a set of silverware for long.

Given that you have a weight limit **W** that you can burgle at one time, and a house that contains various items with weights $\mathbf{w_i}$ and values $\mathbf{v_i}$, devise an algorithm that will net you the most possible money given your weight limit.

Hopefully you recognize this as a recursive backtracking problem; try all possible combinations, taking (or not) each item, and calculate the resulting benefit (returning the overall max). This works, but would take a *long* time for a big house with a lot of items (what would the Big-O for this be?).

Just as in the previous problems, the same sub-problems are constantly being re-solved. Maintain previously solved sub-problems (in an array or a HashMap). When solving the current iteration of the problem, first check if it has already been solved. This type of problem is called a 0/1 knapsack problem. Help can be found on Canvas if you get stuck.

```
    int[] weights = { 6, 1, 2,  5, 4, 3};
    int[] values  = {10, 5, 7, 12, 8, 6};
```

Given a weight limit `W = 10`, your method should return a max benefit of 26.

You will read test cases from a .dat file. The inputs will have the format below.

```
3

6 - 10

6, 1, 2, 5, 4, 3
10, 5, 7, 12, 8, 6

50 - 850

7, 0, 30, 22, 80, 94, 11, 81, 70, 64, 59, 18, 0, 36, 3, 8, 15, 42, 9, 0, 42, 47, 52, 32, 26, 48, 55, 6, 29, 84,
2, 4, 18, 56, 7, 29, 93, 44, 71, 3, 86, 66, 31, 65, 0, 79, 20, 65, 52, 13
360, 83, 59, 130, 431, 67, 230, 52, 93, 125, 670, 892, 600, 38, 48, 147, 78, 256, 63, 17, 120, 164, 432, 35, 92,
110, 22, 42, 50, 323, 514, 28, 87, 73, 78, 15, 26, 78, 210, 36, 85, 189, 274, 43, 33, 10, 19, 389, 276, 312

40 - 100000

2815, 1503, 2916, 9073, 3743, 6124, 9004, 6929, 4795, 9053, 6722, 5634, 5892, 3727, 5113, 8524, 3838, 6962,
7952, 7632, 4831, 9549, 3481, 1879, 1472, 3469, 4641, 1811, 4510, 6414, 1336, 3369, 8018, 3253, 8963, 8486,
6248, 2399, 8477, 1692
4388, 6250, 4556, 5356, 3288, 3917, 7616, 7524, 9769, 3236, 2695, 3524, 7411, 1719, 2480, 9116, 4389, 2392,
1611, 9496, 4140, 7373, 2723, 8654, 5390, 9758, 4262, 8891, 2577, 3643, 6316, 7635, 5006, 1336, 6863, 8826,
2376, 1256, 8691, 2395
```

The first integer represents the number of test cases. The next set of integers separated with a '-' are the number of integers in each set pair followed by the weight limit for that set pair (i.e. set size – weight limit). The first set of integers in the pair is the item weights and the second set is the corresponding item values. An .out file has been provided to check your output.

(Advanced) Word Break

The "word break" problem is as follows:

```
Given an input string and a dictionary of words, segment the input string
into a space-separated sequence of dictionary words if possible. For
example, if the input string is "applepie" and dictionary contains a
standard set of English words, then we would return the string "apple pie"
as output.
```

Use recursive backtracking and memorization to solve this problem. You can test with the below:

```
public static void main(String[] args) {
      WordBreak w = new WordBreak();

      Set<String> dict;
      dict = new HashSet<>(Arrays.asList("hello", "how", "are", "you", "today"));

      System.out.println(w.wordBreak("howareyou", dict));  //how are you
      System.out.println(w.wordBreak("todayhello", dict)); //today hello
      System.out.println(w.wordBreak("helloworld", dict)); //null (no solution)
}
```