CS61B, Fall 2008 **HW #9** **P. N. Hilfinger**

Due: Wed., 3 December 2008

Homework Exercises. You'll find a skeleton for your answers in the `hw9` staff directory.

1. Do exercise 12.1 from *Data Structures Into Java.* However, the template is in `hw9/Chase.java`, rather than what the problem says. Take input to the program from the standard input (System.in).

2. Fill in the implementation of the union-find class in `hw9/UnionFind.java`.

3. In the standard Unix `make` program, the notation

$$F_0 : F_1 F_2 \ldots$$

means both "to create F_0, you must first create or have available F_1, F_2, ..." and also "whenever you change any of F_1, F_2, ..., you must then modify F_0." Each of the F_i is a string of non-blank characters, not including colon. Fill in the template `hw9/Make.java` to read a number of lines in this format and then to generate a list of lines of the form

`Generate` F_i

that will create all the F_i that appear to the left of a colon in the input in such an order as to obey the rules in the input. If this is impossible (due to circular rules), print just

```
Circularity detected.
```

instead.

4. [M. Dynin, from a contest in St. Petersburg] Given a rectangle of letters (such as

```
AB
BC
```

), a starting position within that rectangle (such as the character in the upper-left corner), and a string (such as `"ABBC"`), consider the question of finding paths through the letters that match the given string, begin at the starting position, and at each step move one square in one of the eight compass directions (north, south, east, west, northeast, northwest, southeast, southwest). For the given example, there are two such paths. They are (`E-SW-E`) and (`S-NE-S`)—that is, "one step east, one step southwest, one step east" and "one step south, one step northeast, and one step south." For the rectangle

```
CBB
BBA
```

and the string `"BBCBBA"`, starting at square in the middle of the top row, there are 12 paths. Paths are allowed to visit the same position twice.

Using the template in `hw9/CountPaths.java`, you are to write a program that, given such a rectangle, string, and starting position, reports the *number* of distinct paths that match the string, according to the definitions above. The input (on the standard input, System.in) will consist of four positive integers (call them M, N, r, and c), a string (S), and M strings consisting of upper-case letters `A-Z`, each of which is N characters long. These inputs are all separated from each other by whitespace. The M strings are the rows of the rectangle, from top to bottom. The pair (r, c) are the coordinates of the starting position, with $0 \leq r < M$, $0 \leq c < N$. Row 0 is the top row; column 0 is the left column. The output will be a line reporting the number of paths in the format shown in the examples.

You may assume that the number of paths is less than 2^{31}, $M \leq 80$, $N \leq 80$. Nevertheless, be aware that the time limit is less than a minute.

Example 1:

Input	Output

`2 2 0 0`
`ABBC AB`
`BC`
	`There are 2 paths.`

Example 2:

Input	Output
`2 3 0 1 BBCBBA CBB BBA`	`There are 12 paths.`