**CSCI 3005 – SPRING 2022 – PROGRAMMING ASSIGNMENT 2 – COLOR PALETTES**

A paint shop owner is faced with the problem of displaying N distinct vertical color strips representing all available paint colors at the shop. The most popular colors are to be displayed on the left side of the wall, proceeding with the less popular colors towards the right side. However, there are certain pairs of colors that cannot be placed next to each other for aesthetic reasons. Your basic task is to write software to produce the color arrangement that, for each position starting at the left, selects the most popular unused color that does not conflict with the previous color and that allows all N colors to be displayed.

Given your knowledge of discrete mathematics, you conclude that an exhaustive strategy would involve the generation and testing of N! possible arrangements, yielding a solution that is impractical unless N is very small. Fortunately, you realize that a backtracking strategy will allow you to position the colors in the desired order while discarding color sequences that violate the constraints of the problem. Your assignment is to implement a backtracking solution (either recursively or using a stack) that produces the desired color sequence, if one exists.

Your solution is to be stored in the `PaintShop` Java class, which is expected to have three methods:

`PaintShop(String filename)`: a constructor that reads problem data from the file passed as argument (see example). The first line contains one integer $N$ indicating the number of colors/vertical strips to be displayed. On the second line are $N$ colors (strings) ordered in strictly decreasing order of popularity. The third line contains a positive integer $M$ indicating the number of color conflicts. Each of the next $M$ lines contains two space-separated colors that should not appear next to each other in any order.

```
4
purple pink red black
2
purple pink
red black
```

`String getSolution()`: returns the preferred color arrangement that satisfies the given constraints, in a comma-separated list enclosed by brackets. If no solution exists, the method returns the String NONE. For the file above, the method returns a String containing `[purple, red, pink, black]`

`int howManySolutions()`: returns the total number of arrangements that include all colors while avoiding conflicts, regardless of order. For the sample file, the method returns 8.

You should initially test your methods using the `PaletteTest` class and sample text files provided. The source code for PaintShop.java should be submitted to Mimir. As in previous courses, your program will be graded based on correctness of results, documentation, selection of data structures, design, and style. To encourage development of your program in stages, keep in mind the following milestones:

- To be eligible for C credit, your solution should produce an arrangement that includes all colors while avoiding conflicts, even though the arrangement may be different from what is expected
- To be eligible for B credit, your solution should produce the expected arrangement of colors
- To be eligible for A credit, your solution should produce the expected arrangement of colors and the correct number of solutions

Note that a solution using an approach other than backtracking will earn no higher than D credit.