Homework #3: Color Graph (100 pts)

For this assignment, you must submit **C++** source code that reads in a *directed* graph and outputs **two** items: 1) the **graph structure** and 2) the **number** of unique color combinations between *adjacent* nodes.

Specifically, your code must meet the following requirements:

- *Inputs* **N > 0** lines of input from **stdin**
  - Each line will have the format: **`COLOR-# COLOR-#`**
    - **COLOR** is a color string (e.g., *RED*)
    - **#** is a node identifier (e.g., *4*)
  - Each line represents a *directed* connection, e.g.,
    - **`RED-1 GREEN-2`**
    - indicates a *directed* edge between node "RED-1" and "GREEN-2"

- *Outputs* the **graph structure** with the following format:

```
COLOR-#
        => COLOR-#
        => COLOR-#
        => …
COLOR-#
        => COLOR-#
…
```

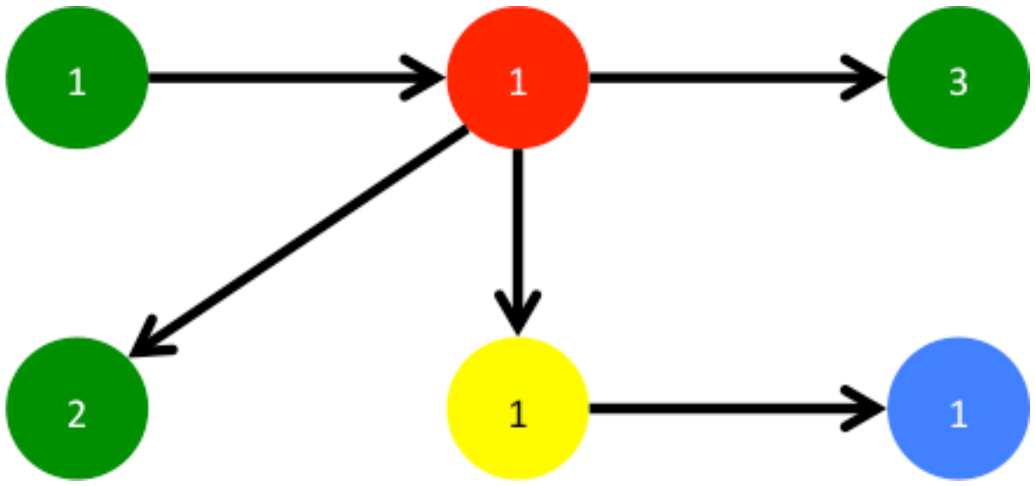
  - The left column represents all "parent" nodes
  - Under each "parent", all of its *adjacent* nodes are listed

- *Outputs* the **number** of *unique* color combinations between adjacent nodes, e.g.,
  - `RED -> GREEN: 6`
  - `RED -> YELLOW: 4`

HINTS:

- while(cin >> str1 >> str2)
- maps
- iterator
- Node struct (or class)
- strings
- string functions
- test, test, test
- file redirection for rapid testing

EXAMPLE:



```
UNIX> cat input.txt
GREEN-1 RED-1
RED-1 GREEN-2
RED-1 YELLOW-1
YELLOW-1 BLUE-1
RED-1 GREEN-3

UNIX> ./a.out < input.txt
======== GRAPH ========
BLUE-1
GREEN-1
     => RED-1
GREEN-2
GREEN-3
RED-1
     => GREEN-2
     => GREEN-3
     => YELLOW-1
YELLOW-1
     => BLUE-1
====== END GRAPH ======

==== COLOR COMBINATIONS ====
GREEN -> RED: 1
RED -> GREEN: 2
RED -> YELLOW: 1
YELLOW -> BLUE: 1

UNIX>
```