COMP 345 Assignment 3 Grading Schema

Marker: Haotao (Eric) Lai
Contact: h_lai@encs.concordia.ca

General Requirement

1. All demos should take place under the Demo Guideline (this is a link), if you didn't read it before I strongly suggest you go and read it.
2. The reinforce, attack and fortify methods created in assignment 2 should be used in part 1. In other words, all the features from assignments 1 and 2 should be working.
3. Official game rules should be followed at all times.
4. No hard-code is allowed, any hard-code program will directly result in zero mark for that part.

Non-implementation Part (8 points)

- Knowledge/correctness of game rules (2 points)
- Incorrect knowledge of official game rules during the demo or incorrect implementation of rules in the presented code will result in mark deduction.
- Modularity/simplicity/clarity of solution (2 points)
  - Data structures should be appropriate, simple and clear. If a team has difficulties explaining their solution, it will be considered unclear.
- Proper use of language/tools/libraries (2 points)
  - .h and .cpp files should be correctly used.
- Code readability (2 points)
  - Improper naming, messy code layout, commented-out code, etc. will result in mark deduction.

Implementation Part (12 points)

Part 1 (4 points)

1. The program must have a obvious structure of Strategy Pattern; (1 points)
2. You should be able to explain which component in your program represents which part in the pattern model;
3. The definition of weakest country is the country with the least number of armies (if you have multiple weakest countries, you can pick one of them to reinforce);
4. Your test driver should be able to clearly demo the human player, aggressive computer player and benevolent computer player; (each kind of player's implementation is 1 point, total 3 points)

Part 2 (4 points)

1. The program must have a obvious structure of Observer Pattern; (1 points)
2. You should be able to explain which component in your program represents which part in the pattern model; (1 point)
3. You should prepare at least two different game scenarios to show your program can output the information dynamically; (2 points)

Part 3 (4 points)

1. The program must have a obvious structure of Observer Pattern; (1 point)
2. You should be able to explain which component in your program represents which part in the pattern model; (1 point)
3. You should provide at least two test cases to show that your statistics will be dynamically updated when the game go further; (2 points)