Before we dive into the requirements, here are a few things you need to know about the logistics of this project:

- Please make a UML diagram of your planned design for each of the client and the server. Set up an appointment to discuss the initial version with your TA no later than **March 5**. Revise them as needed and submit final versions with your final code. This is part of your design grade.
- We expect you to use issues, feature branches, pull requests, and perform code reviews. The process you use for software development is also part of your grade.
- As always, we expect your code to be commented, have good variable names, clean formatting, and well-abstracted methods. Your group should define its own coding standard, and you should all ensure that you follow it.
- Your TA is your "product owner" but also your mentor. They will not have a say in your prioritization for your sprint backlogs; however, you should meet with them regularly—have sprint reviews with them at the end of each sprint within an evolution. There should be no surprises in grading—you should know what your TA thinks of your project all throughout.
- As we discussed in class, if you are having team problems, you should attempt to resolve them yourselves, and if that fails, involve your TA and/or professor. Whether your team is working well or poorly together, we ask that each of you do an *individual contribution assessment* at the end of each evolution. We will post a link to this form later.

List of things you will be graded on:

- Functionality
- Design
- Documentation (including initial UML review + final UML diagrams).
- Testing
- Other Code Quality Factors (*e.g.*, naming, formatting, smells).
- Process (Issue tracking, CI/CD, Code Reviews, etc).
- Team participation

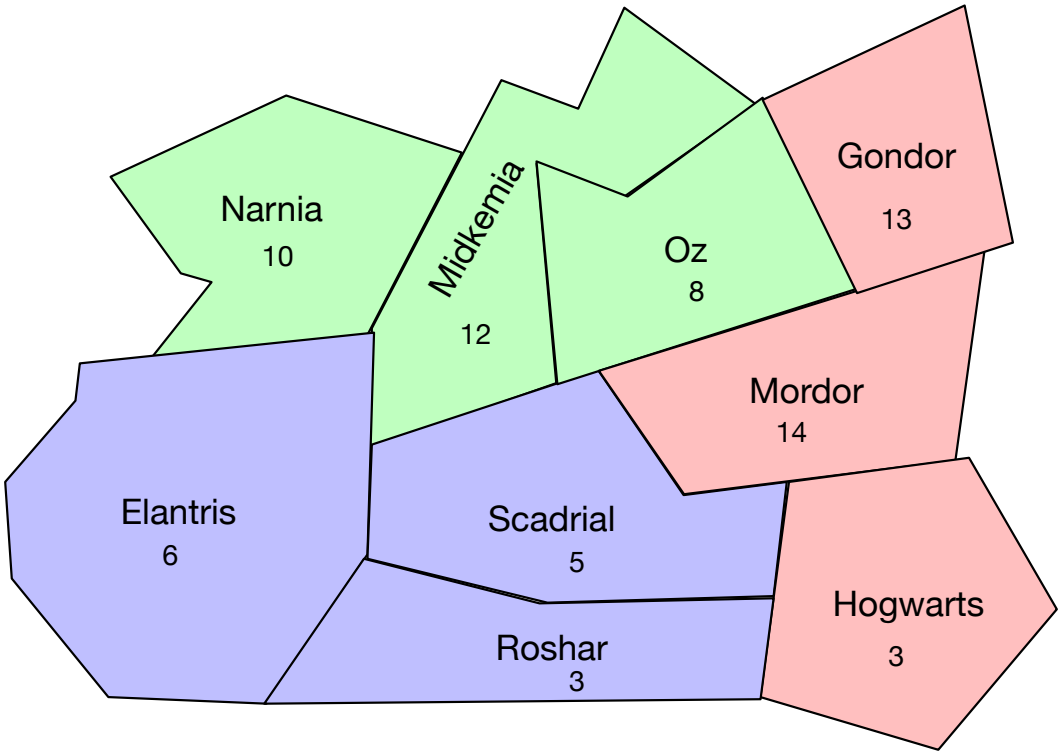
For your team project, you are going to be writing a board game inspired by the board game "RISK," but with very different rules.

We are going to specify many requirements, but leave you a lot of flexibility in exactly what you do with many of them. For example, we are going to specify that you make a text-based client and a server. However, we are not going to tell you exactly the messages to print, prompts to make, how to ask the user to enter input—that is all up to you, as long as what you do is reasonable and usable. Likewise, we are going to tell you to make a set of territories, but are not going to specify what specifically those territories are nor how they are connected.

Before we dive into the specific requirements for Evolution 1, we want to remind you that this project is about **change**—you know that Evolutions 2 and 3 are going to change the requirements. While you don't know what those are, you should try to write code that can handle change well.

The only change we will tell you is that you can expect to write a graphical user interface in a future Evolution (so all of you interested UI/UX design, don't despair of this text-based Evolution). Other than that, we won't say anything about the changes that are coming!

We also want to give you an overview of the game, and define some key terms. First of all, each player's goal is to conquer "the world." This work is composed of a set of *territories*—areas on the map which are each controlled by one player. We'll discuss this game with the following example map that has 9 territories[1]: Narnia, Midkemia, Oz, Gondor, Mordor, Hogwarts, Scadrial, Elantris, and Roshar:



In our example, there are three players: green, blue, and red, and each occupies 3 territories. For example, Narnia, Midkemia, and Oz belong to the green player. Each territory also has some number of *units* in it. In the RISK board game, units are soldiers, however, we are going to use a

[1]For those who are curious, these are named after places from various fantasy books

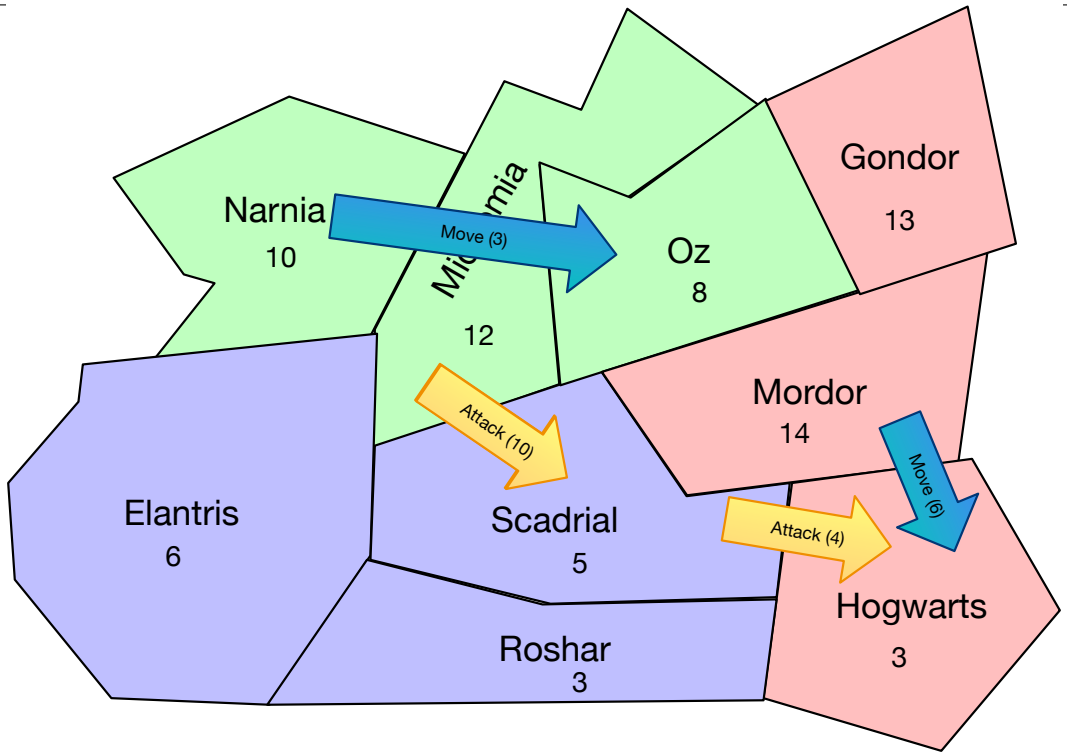
more general term, so that you can imagine any theme to the game that you want, which does not have to be military (if you want to make it a game about, *e.g.*, ants conquering the known world of the back yard, you can, and then units might be ants). In our example, each territory also contains the number of units in it, *e.g.*, Narnia has 10 units while Elantris has 6 units.

Game play consists primarily of two types of actions (at least for this Evolution): *move* and *attack*. A *move* action lets a player relocate units within their own territories. For example, the red player may be concerned that Hogwarts has a very small number of units, so may wish to relocate some units from a stronger territory. They could, for example, issue a *move order* to move 6 units from Mordor to Hogwarts, leaving 8 in Mordor (14-6) and 9 in Hogwarts (3+6). A player may move units where the "from" and "to" are not adjacent, so long as there is a path of their own territories through which the units can move. For example, the green player can move units from Narnia to Oz, even though they are not adjacent since they can go through Midkemia, which is also controlled by the green player.

A *attack* order allows a player to send their units to an adjacent territory controlled by a different player, in an attempt to gain control over that territory. When an attack happens, the attacking units "fight" (again, while we call this fighting, you are welcome to use any theme you want: if your units hold a dance off, baking competition, or any other method of competing that goes with your theme, that is up to you) the defending units. A battle lasts until either the attacker or defender runs out of units participating in the battle, at which point the side with units remaining controls the territory. We spell out the specifics of how attacks are resolved in the detailed requirements. Please note that all the details of resolving combat are very different from the board game RISK.

One major difference between this game and most board games is that all players take their turns at the same time. The players enter 0 or more moves, then *commit their moves*—they say that they are done entering moves for this turn and do not wish to make changes. Once all players commit their moves, the game resolves the outcome of all moves, then reports that to the players, then the next turn happens in the same fashion.

This diagram shows a conceptual representation of what one turn's orders might be:

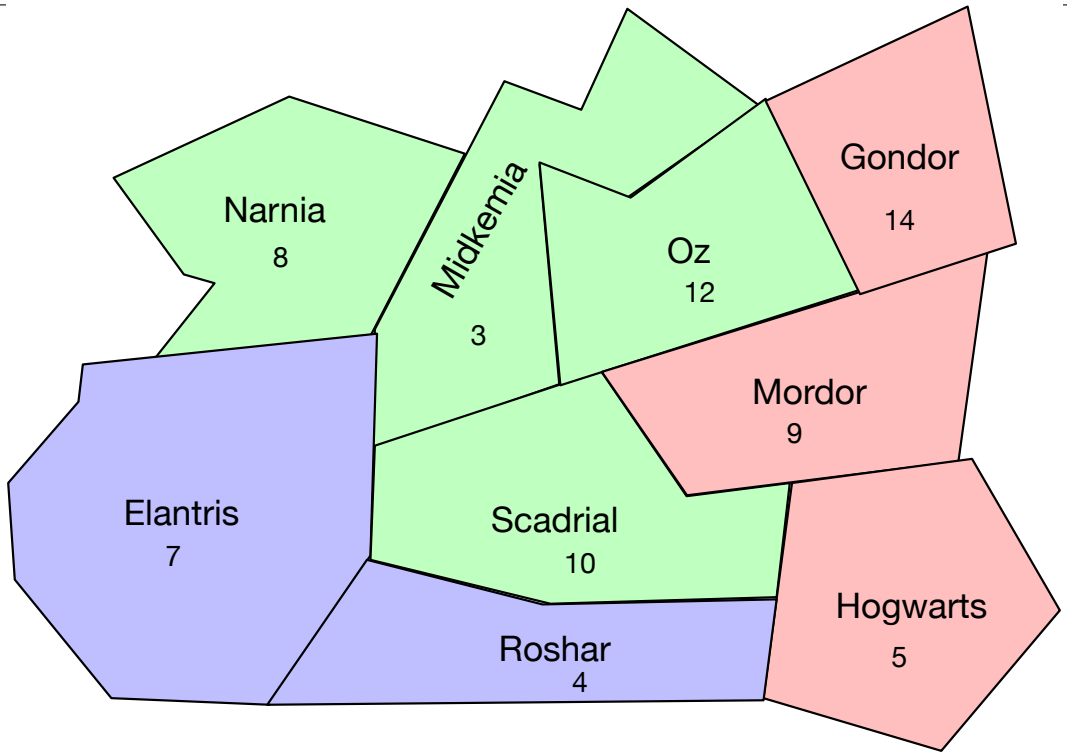


Here, the green player has issued two orders: move 3 units from Narnia to Oz and attack Scadrial with 10 units from Midkemia. The blue player has issued one order: attack Hogwarts with 4 units from Scadrial, and the red player has issued one order: move 6 units from Mordor to Hogwarts.

As you will see in the requirements, move orders happen before attack orders. This ordering means that the battle of Hogwarts will have 4 attackers and 9 defenders (the units moving from Mordor arrive first). The battle of Scadrial will have 10 attackers and only 1 defender, since the other 4 units have left Scadrial to go to Hogwarts.

After all orders are executed, the number of units in each territory increases by 1 at the end of the turn.

Here is one possible situation for the start of the next turn, given the orders in the previous turn:



Green won the battle of Scadrial, and that territory now belongs to the green player. Green attacked with 10 units, lost 1 unit in the battle, but then gained one new unit in each territory at the end of their turn, so green has 10 units in Scadrial. Blue attacked Hogwarts, but lost (red was wise to move reinforcements into that territory). Note that red lost 5 units defending the territory.

Whenever a player has no more territories, that player loses. When one player has all territories, that player wins.

Note that while we gave a graphical description above, your client will be text-based for this Evolution. Accordingly, you might display the state of the game with something like this (the exact format is up to you, but please make sure it is readable and contains sufficient information to play!):

```
Green player:
-------------
 10 units in Narnia (next to: Elantris, Midkemia)
 12 units in Midkemia (next to: Narnia, Elantris, Scadrial, Oz)
  8 units in Oz (next to: Midkemia, Scadrial, Mordor, Gondor)

Blue player:
------------
  6 units in Elantris (next to: Roshar, Scadrial, Midkemia, Narnia)
  3 units in Roshar (next to: Hogwarts, Scadrial, Elantris)
  5 units in Scadrial (next to: Elantris, Roshar, Hogwats, Mordor
                               Oz, Midkemia, Elantris)

Red player:
-----------
  13 units in Gondor (next to: Oz, Mordor)
  14 units in Mordor (next to: Hogwarts, Gondor, Oz, Scadrial)
   3 units in Hogwarts (next to: Mordor, Scadrial, Roshar)

You are the Green player, what would you like to do?
 (M)ove
 (A)ttack
 (D)one
```

With that introduction, here are the specific requirements for Evolution 1:

1. Networked game play.
   (a) 2-5 players should be able to play from different devices at a time. If you would like to enable more, you can.
   (b) There will be a server and a text-based client.
   (c) The server is ultimately responsible for enforcing the rules of the game.
   (d) Clients may validate data in advance, however, the server must do it's own check.
   (e) Note: your server is not required to have state that persists after the server exits. You may have such if you wish, but it is not required.
2. The game board shall be a map, divided into territories.
   (a) Each territory shall be "owned" by one player at any given time.
   (b) Each territory shall have a number of units in it.
   (c) Each territory shall be adjacent to one or more other territories.
   (d) The territories must form a connected graph (all territories must be reachable from any other territory).

(e) Note that the exact map layout and details are up to you.

3. At the start of the game, an initialization phase shall take place where each player selects their starting territories and army placement:

   (a) Territories shall be divided into initial starting groups (with the same number of territories in each group.

   (b) Each player shall pick (or be assigned) one such group as her starting territories.

   (c) Each player shall have the same number of initial units, which she may place in her territories as she wishes.

   (d) The exact number of initial units is up to you, or may be an option in setting up a new game. However, all players must receive the same number of units. Please do make it sensible if you make this a fixed number of units.

   (e) Initial unit placement occurs simultaneously for all players, with no information conveyed between players about the other's places until the process is complete. Put a different way, each player should be able to place their units, then indicate they are done.

   (f) Once all players have finished their unit placement, the normal flow of game play begins.

4. Turn structure: A turn has three parts, which occur in the following order:

   (a) Issue orders.

      i. There are two types of orders that a player may issue: *move* and *attack* (more below).

      ii. A player may issue any number of each type of these orders in a turn.

      iii. A move order must specify the number of units to move, the source territory, and the destination territory.

      iv. Units moving with a move order must have a path formed by adjacent territories controlled by their player from the source to the destination.

      v. An attack order results in units attacking a territory controlled by a different player.

      vi. An attack order must specify the number of units to attack, the source territory, and the destination territory.

      vii. Units may only attack directly adjacent territories.

      viii. A player should be able to indicate when they are done, at which point we say their orders are "committed."

      ix. No player may see the orders of any other player until all players' orders are committed.

      x. The server must ensure that all orders are legal.

      xi. Once all players commit their orders, these orders are executed (next step).

   (b) Execute orders: perform their effects

      i. Move orders move units from one territory to another territory controlled by the same player.

      ii. Move orders effectively occur before attack orders.

      iii. Units moving *out of* a territory *do not* participate in its defense.

iv. Units moving *into* a territory *do* participate in its defense.

v. Attack orders effectively result in all attackers leaving their home territories simultaneously, then arriving at their attack target simultaneously.

vi. A successful attack (see "Combat resolution" below) results in the attacker taking ownership of the territory.

vii. All moves must follow the rules of common sense and preclude cheating: orders may not create new units nor allow a unit to be in two places at once (attacking two territories).

(c) Receive new units: At the end of each turn, one new basic unit shall appear in each territory.

5. Combat resolution: The server should resolve each combat action, informing *all* players of the outcome.

   (a) Combat between one attacker and one defender is an iterative process which ends when one side runs out of units in the fight:

      i. The server rolls two 20-sided dice (one for the attacker, one for the defender). .

      ii. The side with the lower roll loses 1 unit (in a tie, the defender wins).

      iii. The order of evaluation alternates between the highest-bonus attacker unit paired with the lowest-bonus defender unit, and the lowest-bonus attacker unit paired with the highest-bonus defender unit. For example, if the attacker has units with bonuses 15,8,1,0 and the defender has units with bonuses 8,8,3,1, then combat starts with A15 vs D1. Assuming the defender loses, you have now Attacker: 15,8,1,0. Defender: 8,8,3. Next you have A0 vs D8. Assuming the attacker loses, you have now Attacker 15,8,1. Defender 8,8,3, so now you do A15 v D3.

   (b) If player A attacks territory X with units from multiple of her own territories, they count as a single combined force.

   (c) If multiple players attack the same territories, each attack is resolved sequentially, with the winner of the first attack being the defender in subsequent attacks. For example, if A,B, and C attack territoriy X held by player D, then B fights D first. If D wins, then C fights D. If C wins, then A fights C. The sequence in which the attacker's actions are resolved should be randomly determined by the server.

   (d) If units from territory X attack territory Y, and at the same time, units from territory Y attack territory X, then they are assumed to take drastically different routes between their territories, missing each other, and ending up at their destination with no combat in the middle. For example, if all units from X attack Y, and all units from Y attack X, then (assuming no other players attack those territories) both attacks will be successful with no units lost by either side (since there will be no defenders at the start of the battle).

6. Victory and defeat

   (a) A player loses when he no longer controls any territories.

   (b) When a player has lost, he may no longer make any moves, and the server should automatically consider his moves to be committed (as the empty set) at the start of each turn.

(c) A player who has lost may continue to watch the game if he desires, or may disconnect.

(d) A player wins when she controls all territories in the game.

(e) When a player has won, the server should announce this to all remaining clients, which should display this information. The game then ends.

(f) When a game ends, you MAY either have the server exit, OR have it provide the option to start a new game.

Groups with only 3 members may skip all of requirement #3 (unit placement), and may instead simply start the game with a fixed number of units in each territory.