# **Software Engineering Project 1**

# **(Comp 10050)**

## **Assignment 3 – Board representation, Attack Types and Game Logic**

**Aim:** Representation of the board using a linked lists; implementation of different types of attack and the game logic

You are expected to perform this assignment in groups and to use a distributed software repository (GIT).

## **A. Detailed Specification**

The objective for this assignment is to extend the previous assignment with the full game logic. Each game has 3 main phases: preparation, execution and termination.

**Preparation**

* The program asks the user to input a set of players (max 6). For each player the user has to select a type (Elf, Human, Ogre, Wizard) and input a name.
* Each player will be represented as a struct characterised by the fields identifying the player (player type and player name), life points, and the fields characterising the player capabilities (Smartness, Strength, Magic Skills, Luck and Dexterity). The life points are represented as an integer initially set to 100. The capabilities are represented as integers that can assume values in [0, 100] and are randomly assigned to each player following the criteria indicated in section B (same as in the previous assignment).
* Subsequently the program creates a 7 x 7 squared board of slots. Each slot should have references to the adjacent slots.
* The type of each slot is selected randomly. The type of a slot could be: Level Ground, Hill or City.
* Subsequently the players are placed in a slot randomly. Note that, differently from the previous assignment, now more than one player can be assigned to a slot.

**Execution**

* The game can start only if at least 2 players have joined the game.
* The game should be carried out in rounds. At each round a player can decide whether a) to move to an adjacent slot, b) to attack or c) to quit the game . If s/he decide to move, the game should allow the player to select the slot s/he can move to from a set of possible adjacent slots. His/her capabilities will be modified following the criteria specified in section C.
* There are different types of attacks among which a player can choose (See Section D). For each attack type, a player should also be able to decide who to attack, if more than one player can be attacked. Only one attack at a time can be performed for each player’s turn and an attack can only be performed if the player has the requirements to do so.
* When a player’s life points become <= 0 that player has to leave the game.

**Termination**

* The game can terminate only if:
  + Only one of the players remains in the game OR
  + All the alive players decide to leave the game.

## **B. Criteria for Initial Assignment of Capabilities to Players**

* If the player is a Human:
  + all his/her capabilities should be > 0
  + the total sum of the capabilities should be < 300
* If the payer is an Ogre:
  + Magic Skills should be set to 0
  + Smartness <= 20
  + Strength >=80
  + Dexterity >=80
  + The sum of Luck and Smartness should always be <= 50
* If the player is an Elf:
  + All the capabilities should be > 0
  + Luck >= 60
  + Smartness >=70
  + Strength <= 50
  + 50 < Magic Skills < 80
* If the player is a Wizard:
  + All the capabilities should be > 0
  + Luck >= 50
  + Smartness >= 90
  + Strength <= 20
  + Magic Skills >= 80

## **C. Criteria for Modifying a Player’s Capabilities depending on the Slot s/he moves to**

* If the player moves to a Level ground slot
  + Capabilities are unchanged
* If the player moves to a Hill slot:
  + If Dexterity < 50, then the player loses 10 Strength points
  + If Dexterity >= 60, then the player gains 10 Strength points
* If the player moves to a City slot:
  + If Smartness > 60, then the player gains 10 Magic Skills points
  + If Smartness <=50, then the player loses 10 Dexterity points.

## **D. Attacks**

* **Near Attack:**
  + This attack can only be implemented against players who are located in the same slot as the attacker player or in the slots that are adjacent to that in which the attacker player is located
  + If the Strength points of the attacked player are <=70, then attacked player life points = attacked player life points - 0.5 \* attacker player Strength points.
  + If the strength points of the attacked player are > 70, the attacker life points = attacker life points life points - 0.3 \* (attacked player’s strength points).
* **Distant Attack:**
  + It can only be performed against a player who is at a distance > 1 and < 5 from the attacker player.
  + If Dexterity of the attacked player is >= Dexterity of the attacker player, then attacked player life points are unmodified.
  + If Dexterity of the attacker player is > Dexterity of the attacked player, the attacked player life points = attacked player life points - 0.3 \* (attacker player’s strength points).
* **Magic Attack**
  + It can be performed against any player.
  + It can only be performed if (Smartness + Magic Skills) of the attacker player are > 150
  + It determines a decrease in the life points of the attacked player equal to ((0.5 \* Magic Skills of attacker player) + (0.2 \* Smartness of attacker player))

### **Code Design Requirements:**

● Comment your code,

● Use functions where you can.

* Use header files and place function implementations in separate files.

### **Design Hints:**

1. Use array of structs to represent players;

2. Use linked lists to represent slots. Each slot should be identified by a row and a column number. Each slot can have at least 2 up to 4 pointers to its adjacent slots. Create 4 pointers upLeft, upRight, downLeft, downRight pointing respectively to the slots (1,1), (1,7), (7,1), and (7,7).

**Board Slot Example:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |  |
| 6 |  |  |  |  |  |  |  |
| 7 |  |  |  |  |  |  |  |

3. **Use recursion** to retrieve the players that can be attacked in a distant attack.

### **Submission:**

● A doc describing how did you divided the workload in your group. The doc should Include a link to your GIT repository where your project is saved.

* After you submit you should also fill [this questionnaire](https://goo.gl/forms/iGFuZe90mooZLtTH3).