Software Engineering Project 1

(Comp 10050)

Assignment 2 – Players & Slots Representation, Slots and Attack behaviour

Aim: Representation of players and slots types using structs; implementation of slots and attacks behaviour

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

A. Detailed Specification

The objective for this assignment is to create a program that:

- 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.

- Subsequently the program asks the user to select the number of slots (max 20).
- The type of the slot is selected randomly. The type of a slot could be: Level Ground, Hill or City. Slots are stored in an array.
- Subsequently the players are placed in a slot randomly. Note that only one player can be assigned to a slot.

- After all the slots are created, the program asks each users whether s/he wants to move to the next or the previous slot - if possible - or whether s/he wants to attack the closest player.
- If a player decides to move to a slot it loses or gains capabilities according to the criteria described in Section C.
- If a player decides to perform an attack and there are two closest players, s/he should be able to choose between the two of them. The attacked and attacker players can lose or gain points according to the criteria described in Section D.

- After each player has chosen an action, the program prints the name and type of each player and its life points and terminates. These should be formatted as follows: <Player Name>(<Player Type>, <Life Points>. For example:

```
Liliana (Ogre, 50)
John (Wizard, 76)
Evan (Elf, 100)
...
```

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
- City:
    - If Smartness > 60, then the player gains 10 Magic Skills points

    - If Smartness <=50, then the player loses 10 Dexterity points.

D. Implementing an Attack

- If a player (Attacker) decides to attack the closest player (Attacked)
    - If the Strength points of the attacked player are <=70, then attacked player life points = life points - 0.5 * his/her Strength points.
    - If the strength points of the attacked player are > 70, the attacker life points = life points - 0.3 * (attacked player's strength points).

Code Design Requirements:

- Comment your code,
- Use functions where you can.

Design Hints:

1. Use array of structs to represent players;
3. Use array of structs to represent slots. The slots will be considered adjacent to the slots located at the previous index and at the next index in the array.

Submission:

- Submit 3 items through Moodle,
    - A doc describing how did you divided the workload in your group
    - A link to your source code in the GIT repository

After you submit you should also fill this questionnaire.