EE 441 HW#3
Due Date: December 29, 2014 (23:59)

TA's:	**Emin Zerman**	**ARC-202 / DB-14**	**zerman@metu.edu.tr**
	Yeti Ziya Gürbüz	**ARC-202**	**yeti@eee.metu.edu.tr**
	Tayfun Eylen	**EA 404**	**eylen@metu.edu.tr**

*** You may ask HW#3 related questions to eylen@metu.edu.tr**

1. Introduction

As your third homework, you are required to design a turn based multi-player game. The number of players, n, should be selectable and at least 10. There are some layers of players according to their success in previous turns and each player aims to reach the top layer to win the game. In each round, a player's turn is determined randomly. When a player has its turn in a round, two choices exist; fighting with another player or resting for that turn. For fighting case, the player in turn can select the opponent from a set. This set is formed according to the current state of the game, which includes layers of players organized in the form a forest. If a fight takes place, there are three possible outcomes; win, lose or draw. For win or lose case, the layer of the players may increase, decrease or stay unchanged depending on the selected opponent's layer. For the draw case, the player's opponent has a mini-turn with limited options. For resting case, the player increases its success possibility.

2. Layers and opponent selection

Initially, all players start in layer-0. After the first fight, the winner increases its layer to layer-1 and loser stays at layer-0. Moreover, both players are connected to each other within a tree, which determines the opponent set of both players. Another player may join the tree by fighting with a member of that tree. After a while, all players will be in the merged into one tree due to the structure of the opponent set defined by the rules of the game. The aim of the game for a player is being the root of all players for a specific number of turns, which is decided at the beginning of the game as an input.

a. Opponent Selection

As stated above, initially, all players are in layer-0 without having any connection with another player, which means that all players are the roots of their own tree. After some fights, a player can be a root, a leaf or a middle node for that tree.

a.1 Root: If a player is a root, it can only fight with other roots. Other roots may or may not be at the same layer with the player.

a.2 Other Nodes: If a player is not a root, it can only fight with its parent player in its tree.

b. Layers

Fights can change the formation of trees and layers of players. Rules of change are as follows:

b.1 Fights between members of the same tree: Any fight in a tree is done between a parent and a child. If the child wins against its parent, parent and child switch their places in the tree. Any other children of its parent becomes its children and parent of its previous parent becomes its parent (they switch places). If the parent wins the fight, nothing changes in the layers of the tree.

b.2 Fights between roots of trees: After a fight between roots, defeated root becomes a child of the winner root and all its children remain same. If the winning root is in layer-N and loser root is in layer-M, .the formation of layers (tree shifting) is done according to the following rules:

- If N = M + 1; layers of roots and children of both trees stay unchanged.
- If N > M + 1; defeated root's layer becomes layer-(N-1) and all players under the defeated root will be updated accordingly

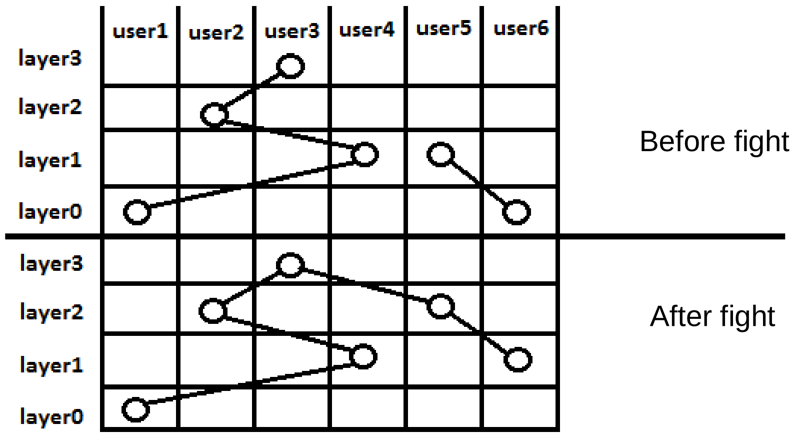

Figure 1: N > M + 1 case (N=3, M=1), left side root wins

- If N < M + 1; winner root is relocated to layer-(M+1) and similar to previous rule all of the tree will be formed with respect to the winner root.

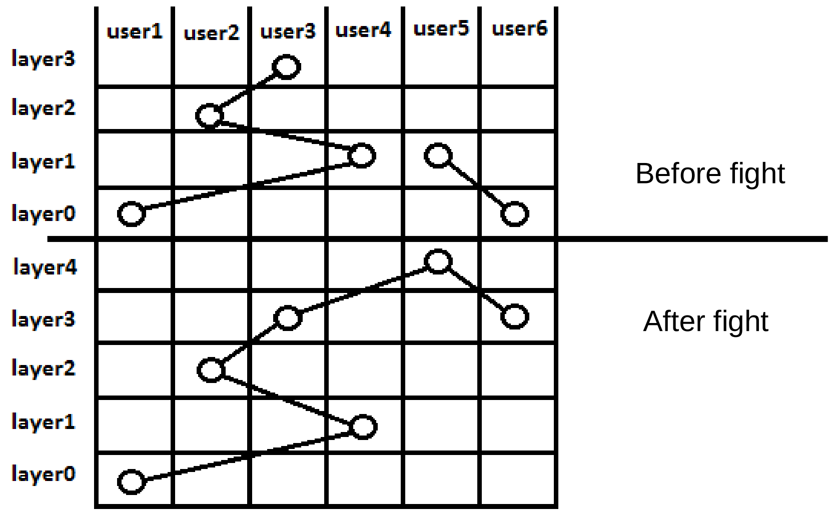

Figure 2: N < M + 1 case (N=1, M=3), right side root wins

c. Randomized turn based

As stated before, this game is a turn based game but there is no static turn sequence for fairness. For every round, every player can play once and the order of play is determined randomly. A uniform random distribution is used for this selection. At the beginning of every round, a turn sequence is determined and announced for that round.

3. Win, lose or draw possibilities and experience points

For any fight between two players, win, lose or draw cases are determined using a uniform random distribution. For this selection process, there exist pre-defined intervals for win, lose and draw. If for a specific player, A, B and C are the intervals for win, lose and draw, respectively. A random number selection is done in interval [0, A+B+C-1]. If the number is in [0,A-1], [A, A+B-1] or [A+B,A+B+C-1], win, lose or draw cases occur, respectively. Before this selection process, A, B and C values should be calculated using 'experience points'. Experience points, E, of players are numerical values used in determining the probabilities to win, lose or draw. It is a function of fights, layers and number of resting moves. There are also a minimum and maximum for experience points. Namely, it is a bounded function whose initial value and the way it changes are explained below:

a. Resting

When a player's turn comes up in a round, fighting and resting are two options for that turn. If resting option is selected, experience points of the player is incremented as follows:

$$E\_player = E\_player + (E\_max – E\_player)/2$$

If a player has already rested in the previous turn, resting is not allowed in the current turn, hence fight being the only option. Namely, successive rests are prohibited.

b. Win

For every win in a fight, a fraction of experience points is added to total experience points of a player. However, according to whose turn it is, coefficients of this ratio changes. Moreover, total experience points of player and opponent of player are also used in this calculation. Let E_player and E_opp be the player and the opponent experience points, respectively. If it is the opponent's turn, which means that the player is challenged with a fight and it has won;

$$E\_player = E\_player + (E\_opp/E\_player)*((E\_max - E\_player)/3)$$

If it is the player's turn and it has won;

$$E\_player = E\_player + (E\_opp/E\_player)*((E\_max - E\_player)/2)$$

c. Lose

Similar to win case, if it is the opponent's turn and the player lost;

$$E\_player = E\_player - (E\_player / E\_opp)*((E\_player - E\_min)/3)$$

If it is the player's turn and it has lost;

$$E\_player = E\_player - (E\_player /E\_opp)*((E\_player - E\_min)/2)$$

d. Draw

Other than win and lose cases, draw may occur in a fight. When draw case occurs, challenged player gains a mini-move or mini-turn, which means that if you are challenged and draw happens, you have a right to choose retreat or fight back. If retreat is selected, nothing changes and the player who has the turn in the first place loses its turn. If fight back is chosen, the player who has the turn in the first place loses some experience points;

$$E\_player = E\_player - (E\_player / E\_opp)*((E\_player - E\_min)/4)$$

E_player: the experience point of the player who has the turn in the first place,
E_opp: the experience point of the player who chose to fight back.

Then, fight is repeated with these new calculated experience points. At this point everything is evaluated as if the second player who is challenged by the first player in the beginning challenges the first one. However, the second player does not lose its move for that turn. If draw happens again after the selection of fight back, two sides retreat and nothing changes.

e. Calculation of win, lose and draw intervals (A, B, C)

Win, lose or draw intervals are calculated by using experience points of players. Other than the effect of layers A, B and C are calculated as shown below;

A = E_player, C = E_opp, B = (E_max – E_min) – |E_player – E_opp|

E_max is 200 and E_min is 100. Initial value of experience point is E_min.

f. Layers

Which layer a player is in does not directly affect experience points; but, it is used to scale the win interval (A). After obtaining E_player (experience points of the player), layer_player (current layer of the player) and E_opp (experience point of the opponent), layer_opp (current layer of the opponent), win interval value of player is;

A = E_player *(1 + (layer_opp – layer_player)/num_of_players)

num_of_players: it is the total number of players, which is taken as an input before game starts.

When fight is over, layer effect is excluded and total experience point of a player is calculated according to the result of the fight. Other than win interval, layers do not affect lose or draw intervals, which means that win, lose and draw intervals are calculated and then win interval is updated with layer effect. If draw case occurs and fight back is chosen, layer effect is excluded from the first player who challenges the second player in the first place. However, the second player's win ratio (or the first player's lose ratio) must be updated with layer effect. As it can be seen from the equation, layer effect increases the possibility of win for lower layer players and or decreases the possibility of win for higher layer players.

4. Examples

```
Enter number of players: 10
How many turns does a player stay in the top layer to win the game: 5
Round 1 turn (move) sequence: 2, 6, 7, 3, 8, 4, 5, 1, 0, 9
Player 2, what's your choice
1.  Fight!
2.  Rest
3.  Look players' status
```

```
3
Choose a player (less than 10)
0
Player 0 is in layer 0 and player 0 has 100 experience points
Player 2, what's your choice
1.    Fight!
2.    Rest
3.    Look players' status
1
Choose an available player (0, 1, 3, 4, 5, 6, 7, 8, 9)
0
Win, lose and draw intervals are: 100-100-100
Random number is 187, draw!
Player 0, what's your choice
1.    Fight back!
2.    Escape
2
.
.
.
Round 4 turn (move) sequence: 6, 4, 3, 8, 9, 0, 1, 2, 5, 7
Player 6, what's your choice
1.    Fight!
2.    Rest
3.    Look players' status
3
Choose a player (less than 10)
4
Player 4 is in layer 3 and player 4 has 157 experience points
Player 6, what's your choice
1.    Fight!
2.    Rest
3.    Look players' status
2
Your new experience point is 175
Player 4, what's your choice
1.    Fight!
2.    Rest
3.    Look players' status
.
.
```

As you can see from the above examples, at the beginning of the game, you should take the number of players and number of moves for win condition as inputs. Then, according to the move sequence all players must play their moves, consecutively. Observing player status is a free option and a player who has the turn can observe any player status more than one. For all fights, results and win, lose, draw intervals should be displayed.

For all requirement stated above, you need to build a player class and a tree structure. <u>**Do not use built in libraries for tree implementation!**</u> To be able to get full grade, implement your classes separately with a header and cpp file.

5. Notice

As an engineering candidate, each student is expected to submit a fully functional homework. In business life, no one would pay for an ill-working program. So, please check your work and debug your program using different inputs and examine corresponding outputs carefully in order to expose any unpredicted or hidden errors in your code. In addition, delivering a product/program with an undesired format harm your reputation in both academic and business life.

You are required to add comments to your code for your colleagues to understand. You may not see this as an important issue; however, it becomes very important if your program exceeds a certain size. You may be working as a programmer next year this time and may be continuing some other programmers' work in your new position. So, please keep your programs comprehensible.

6. Submission

- Homework creation and submission procedures will be covered during the recitation hours. Use Code::Blocks IDE and choose GNU GCC Compiler while creating your project. Name your project as "e1XXXXXX_HW3" where Xs are the digits of your student ID number. Send the <u>whole project folder compressed</u> in a rar or zip file. Name your submission as e1XXXXXX_ee441_hw3.rar. You will not get full credit if you fail to submit your project folder as required.
- Your C++ program should follow object oriented principles, including proper class and method usage and should be correctly structured including private and public components. Your work will be graded on its correctness, efficiency and clarity as a whole.
- You should insert comments to your source code at appropriate places without including any unnecessary detail.
- Late submissions are welcome, but are penalized according to the following policy:
  - 1 day late submission: HW will be evaluated out of 70.
  - 2 days late submission: HW will be evaluated out of 50.
  - 3 days late submission: HW will be evaluated out of 30.
  - 4 or more days late submission: HW will NOT be evaluated.

We wish you success!