BLG 252E - Object Oriented Programming
Assignment #1

Due: April, 12th 23:59

"Luke: I'm looking for a great warrior.
Yoda: Great warrior. Wars not make one great."

- Star Wars: The Empire Strikes Back (1980)

Introduction

For this assignment, you are expected to implement a text-based strategy game called "Great Warriors". First, players will create their characters. Then, they will attack other rulers' lands until they conquer all of the lands or lose theirs.

For any issues regarding the assignment, please contact Doğukan Arslan (**arsland15@itu.edu.tr**).

Implementation Notes

The implementation details below are included in the grading:

1. Please follow a consistent coding style (indentation, variable names, etc.) with comments.
2. You are not allowed use STL containers.
3. You may (and should) implement any getter and setter methods when they are needed.
4. Allocate memories dynamically for arrays and implement necessary destructors where the allocated memory parts are freed. Make sure that there is no memory leak in your code.
5. Since your code will be tested automatically, make sure that your outputs match with sample scenario for given inputs.

Submission Notes

1. Your program should compile and run on Linux environment using "`g++ <your_student_number>.cpp`". You can test your program on ITU's Linux Server using **SSH** protocol. Do not use any pre-compiled header files or STL commands. Be sure that you have included all of your header files.
2. You should compress your files into an archive file named "`<your_student_number>.zip`". Do not include any executable or project files in the archive file. You should only submit necessary files. Also, write your name and ID on the top of each document that you will upload.
3. Submissions are made through **only** the Ninova system and have a strict deadline. Assignments submitted after the deadline will not be accepted.
4. This is not a group assignment and getting involved in any kind of cheating is subject to disciplinary actions. Your homework should not include any copy-paste material (from the Internet or from someone else's paper/thesis/project).

1 Implementation Details

You are expected to implement 3 classes; **Land**, **Character**, **CharacterList**.

1.1 Land

Private Attributes:

Land class *has* a **name** (eg. Gotham) and a **holding** (eg. city).

Methods:

1. **Constructor(s):** the constructor should *optionally* take the name and the holding attributes. Also, you need to implement the copy constructor.

Warning: If you need any other methods for Land class, you are expected to implement them as an inline function.

1.2 Character

Private Attributes:

Character class has a **name** (eg. Alexander), **manpower** (number of soldiers), **gold** (amount of gold), **numOfLands** (number of land owned), and **lands** (head of Lands linked list, the character has).

Methods:

1. **Constructor(s):** the constructor should *optionally* take the attributes. Also, you need to implement copy constructor and destructor for this class.
2. **getTaxes:** method to collect taxes for the character. For every land the character owns, he/she gets gold according to this; 5 golds for each village, 7 golds for each castle, and 10 golds for each city.
3. **addLand:** method to add a land to the character.
4. **removeLand:** method to remove a land from the character.

1.3 CharacterList

Private Attributes:

A CharacterList has **characters** (array of characters in the game) and **size** (number of the characters in the array) attributes.

Methods:

1. **Constructor(s):** implement default constructor and destructor for this class.
2. **addCharacter:** method to add a character to the array.
3. **delCharacter:** method to delete a character from the array. Characters without a land must be deleted from the list.
4. **getPlayer:** method to get reference to the player.
5. **getLandOwner:** method to get owner of the specified land based on lands' name.

Notice: Check the gameplay **here** and implement other necessary functions accordingly, eg. to print menus or getter and setters. Also, be aware that some parts of the necessary functions are already provided to you in main.cpp file.

Other Necessary Methods:

1. **listCharacters:** method to print all characters' name to the console. It should be a **friend function** of CharacterList class.
2. **listLands:** method to print all lands' name, holding and the owner to the console. It should be a **friend function** of CharacterList class.

In the main function, first, read **characters.txt** file to create characters and lands of the game. Store characters in a CharacterList object. Then, wait for the player to create his/her character. The player should start the game with 1 village, 3 manpower, and 20 golds. Also, you should create a general for the player which is a **constant Character object** (Do not forget to implement necessary functions for constant objects). The in-game menu consists of three choices that make a turn pass: "stay in the palace", "attack to a land", and "buy manpower". At the end of each turn, you should call endOfTurn function to collect taxes, check for **rebellions** (occurs when the player has no manpower and as a result, the player loses one of his/her lands), and the **amount of gold** (player loses manpower when has not enough gold to feed them, maintaining 1 manpower costs 1 gold). When the player has no lands or there are no characters left except the player, the game ends. The possible actions in the game are as follows:

1. **Stay in palace:** Do nothing. Go to the next turn.
2. **Attack to a land:** List lands. Ask for a land name to attack. If the player has less manpower than attacked side, the player loses the war and all of manpower. Otherwise, the player conquers the land, losing manpower as much as the manpower of the attacked side (If player has 20 manpower and attacked has 12 manpower then player will have 8 manpower at the end of the war). Show outcome of the war. Go to next turn.
3. **Buy manpower:** Get the manpower order. Buying 1 manpower costs 5 golds (maintaining costs 1 gold). Go to next turn.