MIDDLE EAST TECHNICAL UNIVERSITY, NORTHERN CYPRUS CAMPUS
CNG242 Programming Language Concepts – Spring 2023 – Assignment 3

Date handed out: May 25, 2023, Thursday
Submission due: June 8, 2023, Thursday

Please Read This Page Carefully

Submission Rules

1. You are going to work in groups of two.
2. **You need to write a comment on the first line of your file**, stating that you read the rules specified here and the submission is your own work. **Submissions without this statement will not be graded. Example,**
```
/* Zekican Budin - 1234567
I read and accept the submission rules and the important section specified in assignment file. This is my own work that is done by myself and my team-mate only */
Each person is responsible separately to write the statement on the parts they have completed.
```
- Please refer to the syllabus[1] provided for CNG 242 for the measures in place in case of any academic dishonesty[2,3].
3. You need to be ready to demonstrate your own work, answer related questions, and have short coding sessions if necessary.
4. You cannot share this worksheet with any third parties. Upon doing so, any detected action will directly be sent to the disciplinary committee.
5. The assignment **should not be shared publicly in any manner, at any time.** The assignment cannot be disclosed or disseminated to anyone before, during, or after the submission.
6. You should read the questions fully and follow the directions listed. Only the **functions, operators, classes and/or structures with the same name will be graded.**
7. You can only get full marks, if your files fully compile, run, and generate correct output for all possible cases. **Non-general, static solutions will not be graded!**
8. You are not limited with any libraries. However, **do not use following** keywords; allignas, allignof, asm, auto, char8_t, concept, consteval, constexpr, constinit, const_cast, co_await, co_return, co_yield, dynamic_cast, elifdef, elifndef, export, extern, explicit, goto, import, inline, module, mutable, reflexpr, register, reinterpret_cast, requires, static_assert, static_cast, synchronized, thread_local, unsigned, volatile.
9. Read the important section (last page) under the grading part for submission format and information.

[1] Page 3&4 (Course rules, #1,2,3)
[2] Taking unfair advantage in assessment is considered a serious offence by the university, which will take action against any student who contravenes the regulation through negligence or deliberate intent.
[3] For a comprehensive cheating definition, please refer to: https://ncc.metu.edu.tr/res/academic-code-of-ethics . When a breach of the code of ethics occurs (cheating, plagiarism, deception, etc.), the student will be added to the BLACKLIST.

Middle East Technical University
Northern Cyprus Campus

CNG242 Programming Language Concepts
Assignment 3
Heroes of the Board: Legendary two-player board game

Learning Outcomes: On successful completion of this assignment, a student will:

- Combine previously learned concepts to form a project
- Appreciate clever design and reusability of functionality

In this assignment, you are going to work in groups of two.

The Game

Strategy games are fun to play mainly because they are based on the players' decision-making skills, requiring internal decision-tree-style thinking and good degrees of situational awareness. On the other hand, board games (even relatively simple ones) are quite popular, particularly as mobile applications. According to a set of rules, they involve counters or pieces that are moved or placed on a specifically designed surface or "board". Most board games would also require strategic skills, as explained above. Some of the well-known strategy board games are as follows: Mighty Party[4], Dwarfs fight[5] and DuelystII[6].

In this assignment, you will implement a board game called "**Heroes of the Board: A legendary two-player board game**". In the game, two players (the computer will not play the game) will have one board each. In each square of the board, the players can place their pieces as shown in Figure 1.

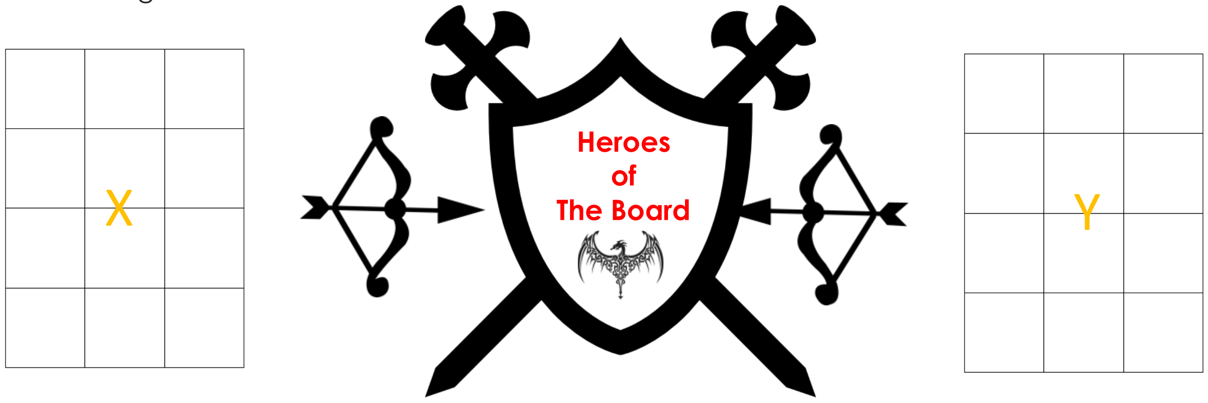

Fig.1 The game board

- With each turn, the players will mark one cell on their board to add a piece. Following this, each piece is going to perform an attack through the row they belong to.
- In case there are no pieces on the way the attack will directly hit the player and the player will lose as much health as the value of the attack. In case any of the player's health is less than or equal to zero, the other player would win the game.
- In case there are pieces in the direction of attack, the pieces on the way will be hit losing health and dying if the health becomes less than or equal to zero.

[4] https://mighty-party.com/
[5] https://nodstudio.itch.io/dwarfs-fight
[6] https://duelyst2.com/

The boards of players have four rows and three columns. Each player has 100 life at the beginning. There are two kinds of pieces Ranged and Melee. Some of the pieces can only be used a restricted number of times on the board. Also, while the Melee pieces (Spearman, Ranger, Swordsman and Knight) can be located in columns 2 and 3, the Ranged ones (Bowman, Elf and Mage) can only be located in columns 1 and 2 (see Figures 2-5). Also, when the **Mage** plays a round, it should give a +1 life point bonus to all its teammates on the board, while when the **Knight** plays a round it gives a +1 attack bonus to all its teammates.

The Ranged pieces are further divided into three categories as follows:

Piece	**Life**	**Damage per hit**	**Character to represent it on board**	**The number of times pieces can be used by a player**
Bowman	12	3	B	Unlimited
Elf	8	6	E	3
Mage	4	8	M	2

There are four types of Melee pieces as follows:

Resource	**Life**	**Damage per hit**	**Character to represent it on board**	**The number of times pieces can be used by a player**
Spearman	6	3	S	Unlimited
Ranger	8	6	R	2
Swordsman	6	4	W	Unlimited
Knight	10	10	K	1

Each player can position one piece in each turn. Let us now consider the scenario given in Figure 2. It is the turn of Player X. Player X has two pieces. A Bowman in A2, an Elf in B2. Player Y also has three pieces. A ranger at A3, a Knight at A2, and an Elf at B2.

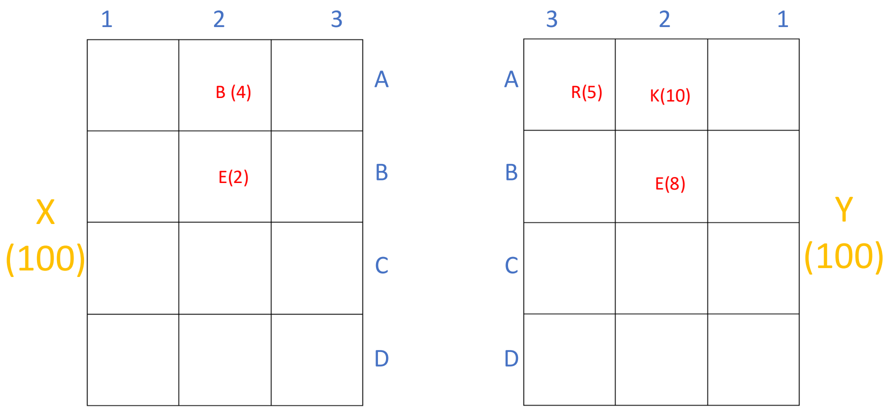

Fig.2 Sample Scenario - 1

Now let us assume that Player X is placing a knight at C3 as shown in Figure 3.

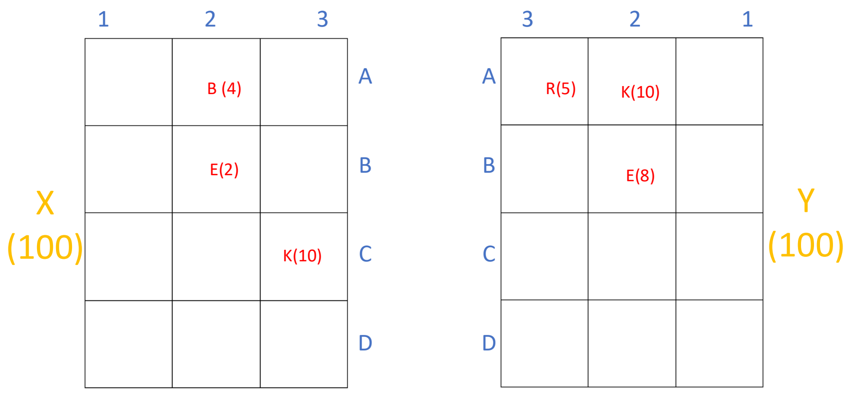

Fig.3 Sample Scenario - 2

At the end of this round, the Bowman of X (A2) will hit the Ranger of Y (A3) since it is at the front. Then the Elf of Player X (B2) will hit the Elf of Player Y (B2). Finally, since there are no pieces in front of the Knight of Player X (C3) it will directly hit Player Y causing damage. So, it means that if it is Player Y's turn, then Player's Y pieces will attack. However, in case of there is no piece in front of Player Y to protect it, then Player Y can be hit by a piece of Player X. At the end of the round the board will look like illustrated in Figure 4.

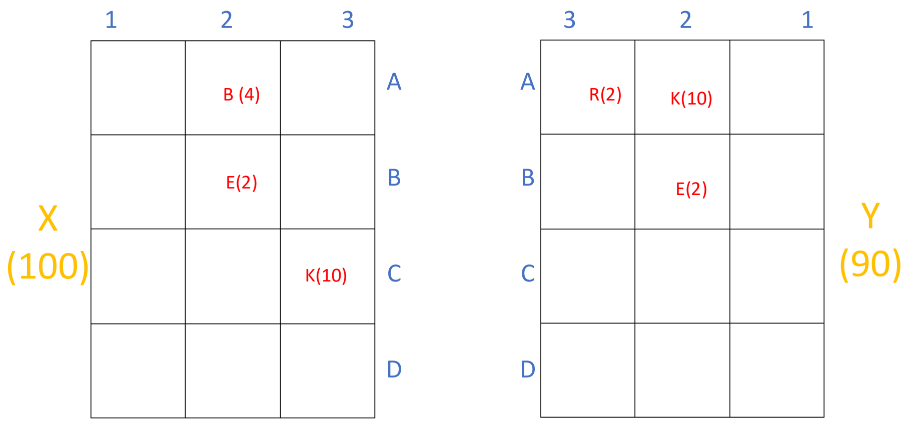

Fig.4 Sample Scenario 3

Implementation

Create a class for the boards. There should be two objects of type board for each player. Create a base abstract class, "Pieces", which will have a life and representing character. There are two sub-classes which are Ranged and Melee pieces. The Ranged class is further derived for Bowman, Elf, and Mage whereas Melee class is derived for Spearman, Ranger, Swordsman, and Knight. You should store the number of times each piece was used for the restrictions identified above.

Inplay Turns

Each player is going to move in turns. Starting player at each turn should be decided randomly. **Before and after choosing the coordinates, the last version of the boards will be shown.** The program should not allow the players to move an already filled coordinate.

If one of the players (X or Y) reach 0 or less health the game will end, and the program will show the details of the kills performed by each player.

Grading and Division of Tasks

Item	Mark	The programmer
Board Class	40	**Student 1**
Pieces Classes	100	**Student 2**
Game Playing	60	**Student 1**

Important (Read Here!):

1. Code modularity, pointer operations, clarity, comments will also be graded. Modularity refers to the class design, how easy it is to create a similar but different game with the developed classes or how easy it is to add a new feature and so on. Hard coding game variables may cause in severe grade loss.
2. The students should work together for identifying the needs of the program.
3. All groups should be ready for demonstration after the final exam period.
4. Please note that the fair contribution of partners is very important for us. Each student will be evaluated for individual contributions, and demonstration sessions will be conducted accordingly, if necessary. In case of failure to demonstrate their own parts, the related student will be penalized following the guidelines in our syllabus.
5. Submissions without the **submission rule** statement will not be graded.
6. If you use any library, function, class that you see online in your project, make sure to understand it fully because you might be asked to demonstrate a different implementation with the same library, function and/or class.
7. Zip your readme.txt file, .exe, .h and .cpp files (i.e. header files and/or C++ implementation files) and submit through ODTU-CLASS. Readme.txt file should include following the following:
   - Used IDE/Compiler name-version with used compiler options (if any)
   - Changes that you have done to project settings (if any)
   - Tested Platform (Windows 10, MacOS, Kali, Ubuntu, …)
   - Bugs that you have noticed (if any)
   - Preferred dates with time intervals for demo session (Must be after your final exam, provide at least 3 dates and time intervals of 2 hours before June 22)
   - Comments about your implementation (if any)
8. Name your zip file with your student id numbers such as: 1234567_2345678.zip