CSCI 1300 CS1: Starting Computing
Instructor: Fleming/Wong, Spring 2019

Project 3: Choose Project, Meet with TA/CA to go over design
Before Wednesday April 10 at 6 PM

Project 3: Submit Class files & Code Skeleton
Due Wednesday April 10 by 11 PM (no early submission bonus)

Project 3: Final Deliverables - Due Wednesday April 24 by 11 PM (no early submission bonus)

Project 3: Project Report - Due Sunday April 28 by 11 PM (no early submission bonus)

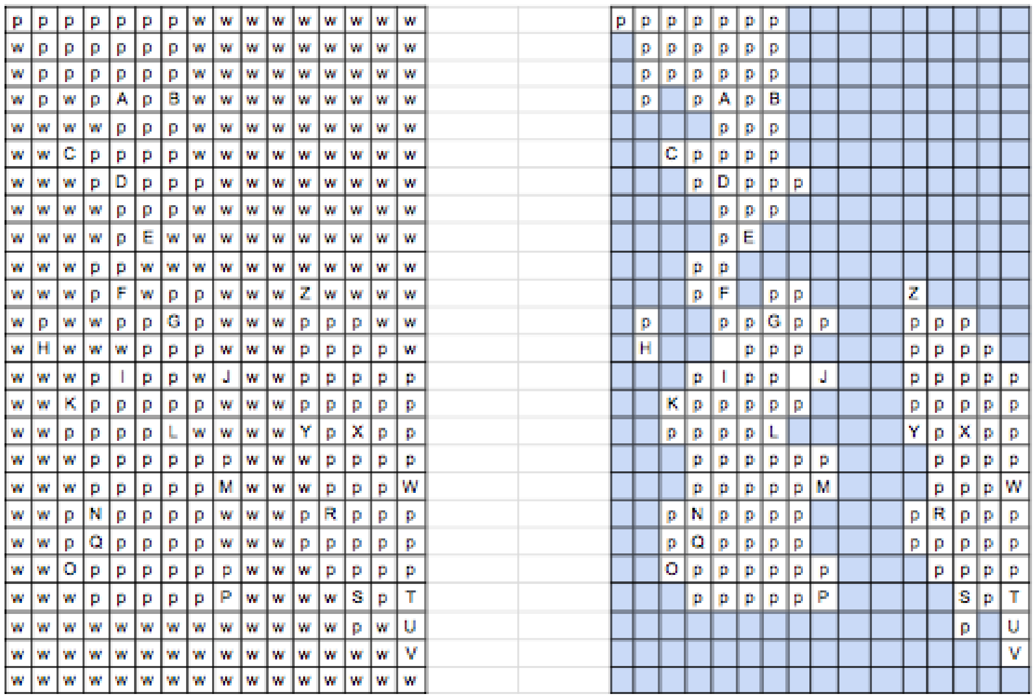
Project 3: Interview Grading
Begins on Thursday, April 25. Must be completed at the latest on Friday May 3.

This project is worth 15% of your overall course grade.

The minimum requirements for the project can be found at the end of the project description.

"When You Play The Game Of Thrones, You Win Or You Die."

For the Final Project we will implement an interactive board game named "**The Game of Thrones**". Someone stole our project idea back in the 90s and wrote a bunch of books and TV shows about it, but whatever. The user plays the role of a hero in a quest through the vast kingdom of ***Westeros*** to defeat the other heroes and sit alone atop the Iron Throne. The kingdom is represented by a board that is a grid consisting of 25 rows by 16 columns of tiles, where each tile is a location in Westeros. The map can be found in the file `mapGOT.txt`.



p	p	p	p	p	p	p	w	w	w	w	w	w	w	w	w
w	p	p	p	p	p	p	w	w	w	w	w	w	w	w	w
w	p	p	p	p	p	p	w	w	w	w	w	w	w	w	w
w	p	w	p	A	p	B	w	w	w	w	w	w	w	w	w
w	w	w	w	p	p	p	w	w	w	w	w	w	w	w	w
w	w	C	p	p	p	p	w	w	w	w	w	w	w	w	w
w	w	w	p	D	p	p	p	w	w	w	w	w	w	w	w
w	w	w	w	p	p	p	w	w	w	w	w	w	w	w	w
w	w	w	w	p	E	w	w	w	w	w	w	w	w	w	w
w	w	w	p	p	w	w	w	w	w	w	w	w	w	w	w
w	w	w	p	F	w	p	p	w	w	w	Z	w	w	w	w
w	p	w	w	p	p	G	p	w	w	w	p	p	p	w	w
w	H	w	w	w	p	p	p	w	w	w	p	p	p	p	w
w	w	w	p	I	p	p	w	J	w	w	p	p	p	p	p
w	w	K	p	p	p	p	p	w	w	w	p	p	p	p	p
w	w	p	p	p	p	L	w	w	w	w	Y	p	X	p	p
w	w	w	p	p	p	p	p	p	w	w	w	p	p	p	p
w	w	w	p	p	p	p	p	M	w	w	w	p	p	p	W
w	w	p	N	p	p	p	p	w	w	w	p	R	p	p	p
w	w	p	Q	p	p	p	p	w	w	w	p	p	p	p	p
w	w	O	p	p	p	p	p	p	w	w	w	p	p	p	p
w	w	w	p	p	p	p	p	P	w	w	w	w	S	p	T
w	w	w	w	w	w	w	w	w	w	w	w	w	p	w	U
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	V
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

The board is comprised of 400 tiles as follows:

- 238 water tiles (marked with a ‘w’),
- 136 land tiles (marked with a ‘p’)
    - 5 random locations of these land tiles hold the resource *dragonglass*.
- 26 land tiles where cities are located (marked with capital letters from ‘A’ to ‘Z’).
    - The names of the cities and more information about each city can be found in the file `citiesGOT.txt`.

The user can choose to play **The Game of Thrones** as one of 6 provided heros, or they can create their own hero. The file `heroesGOT.txt` has information about each of the 6 heroes, in the following format:

```
<Name>,<$>,<influence>,<army_size>,<w1>,<w2>,<w3>,<w4>,<r>,<c>,<ship>
Cersei Lannister,8000,300,300,Jaime Lannister,
                                   The Mountain,NULL,NULL,16,7,no
Stannis Baratheon,1000,100,200,Melisandre,
                                   Davos Seaworth,NULL,NULL,14,9,no
Jon Snow,100,50,20,Jon Snow,Samwell Tarly,NULL,NULL,4,5,no
Sansa Stark,1000,100,200,Brienne of Tarth,NULL,NULL,NULL,7,5,no
Euron Greyjoy,1000,50,100,NULL,NULL,NULL,NULL,13,2,yes
Daenerys Targaryen,100,1000,50,NULL,NULL,NULL,NULL,18,16,no
```

- At the beginning of the game, each hero has at their disposal a starting sum of money and an army of knights, which they will use to find and defeat the other heroes.
- Each hero also has a value for `influence`. This metric speaks about each hero’s ability to build support for their cause. Having a high `influence` helps a hero strengthen their faction by winning new warriors on their side.
- Each hero has a starting location on the map, represented by an integer row-column pair: `<r>`, `<c>`. These are referenced relative to the upper-left corner. For example, `Jon Snow` starts at row 4, column 5, which corresponds to the tile marked ‘A’ on the map which, according to the file `citiesGOT.txt`, is the city of Castle Black.
- Heroes can have between as few as 0 and as many as 4 warriors. Some heroes start with warriors, while others do not. The warriors who start with a hero are represented by names in the `w1` and `w2` spots. As the game progresses, heroes may acquire other warriors, filling up slots `w1` through `w4` sequentially. A value of `NULL` means that warrior slot for that hero is empty. Jon Snow is both a hero and a warrior.
- The `ship` characteristic of each hero refers to their ability to move to ‘water’ locations on the map. Only the hero `Euron Greyjoy` starts with a ship. No other heroes will have the ability to move onto ‘water’ locations unless they acquire a warrior with a ship. Hero `Stannis Baratheon` starts with `Davos Seaworth` as one of his warriors. Since `Davos Seaworth` has a ship, `Stannis Baratheon` also has the ability to move to ‘water’ locations.
- The first five heroes start on the same continent, while the last hero `Daenerys Targaryen` starts alone on another continent.

There are 14 warriors. Each warrior has several characteristics: *strength*, *loyalty*, and *morale*. The values for each warrior are provided in the file `warriorsGOT.txt`, in the following format:

```
<Name>,<strength>,<loyalty>,<morale>,<free>,<ship>,<has_dragonglass>
Brienne of Tarth,100,100,50,no,no,no
Bronn,70,10,80,yes,no,no
Samwell Tarly,10,100,50,no,no,yes
Jon Snow,100,100,50,no,no,no
Asha Greyjoy,60,100,50,yes,yes,no
Melisandre,80,50,50,no,no,no
Petyr Baelish,20,0,100,yes,no,no
Jaime Lannister,120,90,40,no,no,no
The Mountain,200,90,50,no,no,no
Arya Stark,200,100,60,yes,no,no
Davos Seaworth,10,100,40,no,yes,no
Tormund Giantsbane,120,70,70,yes,no,no
Grey Worm,120,100,30,yes,no,no
Tyrion Lannister,50,100,60,yes,no,no
```

- 7 warriors start the game attached to a hero (`free = no`)
- 7 other warriors are roaming free in the kingdom of Westeros (e.g. warrior `Bronn`).
- The only 2 warriors who start with a ship and who have the ability to move onto 'water' tiles are `Asha Greyjoy` and `Davos Seaworth`.
- The only warrior who has *dragonglass* at the beginning of the game is `Samwell Tarly`. Think about *dragonglass* as a weapon each warrior might have. Having *dragonglass* will greatly increase a warrior's chances of defeating the *White Walkers* in the final battle of the game. As each hero (and their warriors) travel the map, they can discover the *dragonglass* that was scattered at the beginning of the game and equip one of their warriors with it.
- You may define additional characteristics for warriors or heroes, as needed.

The values for *money*, *influence* and *army size* for each hero, and the values of *strength*, *loyalty* and *morale* for each warrior evolve during the game, based on the travels and encounters of each character. *Morale* and *loyalty* must remain in the 0-100 range, and *strength* must remain in the 0 to 200 range.

As the heroes travel to different locations on the map, they win more money, and increase the size of their armies. The heroes take control of every place they visit, which brings them points in the game. Controlling a 'water' location is worth **1 point**. A 'land' location is worth **2 points**. Controlling a city is worth between **3** and **30 points**. If a hero would be able to control the entire map, the total number of points would be **800 points**:

- 238 points from controlling all the 'water' tiles.
- 272 points for controlling all the 'land' tiles.
- 290 points for controlling all cities.

The file `citiesGOT.txt` contains information about each city. The values for money, army and points a hero can gain by taking control of each city are specified in the file in the format below:

```
<Letter>,<Name>,<$>,<army_size>,<points>
A,Castle Black,15,15,3
B,Eastwatch by the Sea,15,15,3
C,Deepwood Motte,15,15,3
D,Winterfell,75,75,15
E,White Harbor,30,30,6
...
Z,Braavos,50,50,10
```

The user **wins the game** by defeating the *White Walkers*, a tribe of lovable scamps who are on a mission to murder everyone in the kingdom and turn them into yetis. As a hero starts increasing their army and controlling more territory, their chances of having to face the *White Walkers* also increase. More details about the *White Walkers* and the end-game conditions are given later in the project description. The user's goal is to have as many points as possible when they finally battle - and hopefully defeat - the *White Walkers*. To earn points, players must find and defeat the other heros in the kingdom, conquer territory and amass a large army, because that's how mafia works.

Fig 1. I never learned how to shrug right.

"Never forget what you are, for surely the world will not. Make it your strength. Then it can never be your weakness. Armour yourself in it, and it will never be used to hurt you."

Start the game by loading the text files to set up the game, initializing all the heroes and warriors, and placing *dragonglass* in 5 random locations. If a warrior is attached to one of the heroes, their location will <u>always</u> be the same as the hero's. If a warrior is roaming free in Westeros, assign them a random starting location on the map. Free warriors with ships must start on a 'water' tile, and free warriors without ships must start on 'land' tiles. Free warriors cannot start in the same location as one of the heroes or already-initialized warriors.

Update the heroes' points at the beginning of the game by assigning them the point value associated with their starting location (values can be found in file `citiesGOT.txt`).

Welcome the user and ask for their name. Then, ask them to choose a hero or create their own:

```
Welcome to the G A M E of T H R O N E S!

State your name, Your Grace: George Martin

Would you like to choose a hero George Martin, Your Grace?(Y/N):Y
Please choose from the following heroes:

   1. Cersei Lannister
   2. Stannis Baratheon
   3. Jon Snow
   4. Sansa Stark
   5. Euron Greyjoy
   6. Daenerys Targaryen
```

If the user answers "no" (**N**), ask the user to create their own hero:

```
Your Grace George Martin, you have chosen to go on this journey alone.
Please provide starting values for your adventure.
```

The user should then be prompted to pick starting values for their hero's *money*, *influence*, *army size*, and starting *location*. Provide appropriate value ranges for each choice:

- The starting values for *money*, *influence* and *army size* can be between 0 and 10000
- For starting location, the user has to choose a valid row value (1-25) and a valid column value (1-16).
- If the hero's chosen starting location is on 'water', equip the hero with a ship; they will have from the beginning the ability to move to 'water' locations on the map.
- The location cannot be the same as one of the other 6 heroes or as one of the already-initialized warriors.

Now is the time to show the map to the user. This occurs at the beginning of each turn. The user has visibility in a 7 x 7 area centered at their hero's current location. For example, for main hero "`Cersei`" the board displayed at the beginning of the game would look like this:

~	~	*	*	*	~	~
*	I	*	*	~	J	~
*	*	*	*	*	~	~
*	*	*	L	~	~	~
*	*	*	*	*	*	~
*	*	*	*	*	M	~
N	*	*	*	*	~	~

Where `'L'` is the city of King's Landing, which is the starting location for main hero "`Cersei`". All water tiles should be displayed with the symbol `'~'` and all land tiles with the symbol `'*'`, and and towns are represented with their character initial. If the hero is near/at one of the edges of the map, you may use other characters to represent the edges/outside of the map region. The locations of *dragonglass* should remain hidden to the user until they stumble upon them.

“Fear Cuts Deeper Than Swords.”

Every turn the main hero has 3 choices:

1. **Travel:** ask the user to pick a travel direction: North, South, East or West.
    - If the desired location is a “water” tile, the hero cannot travel there unless they have a ship. If they don’t have a ship, ask the user to choose again.
    - If by choosing a certain direction the new location would be outside the boundaries of the map, ask the user to choose again.
    - If a hero chooses to travel, then they may not stay in the same location

2. **Rest:** this can be a useful option if the user’s strategy is to avoid encountering other heroes, or to strengthen their hero’s party
    - the hero’s location remains unchanged.
    - each warrior in the hero’s party increases their strength by 1

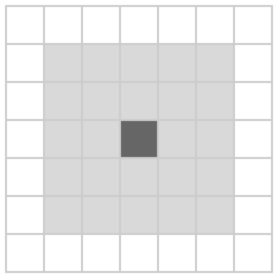
3. **Consult with the gods:**
    - the hero’s location remains unchanged.
    - each warrior in the hero’s party increases their morale by 1

After the main hero’s location is updated based on their choice, proceed to update the location of all the remaining active heroes and free warriors, by randomly moving them one tile, in the North, South, East or West direction. The same rules apply:

- If the generated random location is a “water” tile, the hero cannot travel there unless they have a ship. If they don’t have a ship, generate another random travel direction.
- Heroes must remain within the boundaries of the map.
- Non-main heroes may not stay in the same place; they must move in *some* direction.

“A Lannister Always Pays His Debts.”

After the locations have been updated, it’s time to resolve possible encounters. First, check if the main hero is in proximity to other heroes. If another hero’s location is within 2 tiles in any direction (including diagonally) of the main hero’s location (depicted by the shaded region below), then the two heroes will have an encounter.



If such an encounter occurs, then present the main hero with 3 options:

1. **Give a speech:** the main hero (the user) relies on their *influence* to win over the other hero's army. The main hero's probability of winning the encounter is higher/lower depending on the *morale* of the first warrior in the opposing hero's party. *Note:* The sum of the *influence* values for all the heroes at the start of the game is **1600**.

    a. Compute the probability the main hero will win the encounter as:
    $\left(1-\frac{opposing\ hero's\ 1st\ warrior\ morale}{100}\right)*\frac{main\ hero's\ influence}{1600}$

    Example: Daenerys has an *influence* value of 1000. She is encountering Stannis, whose first warrior Melisandre has a *morale* value of 50. The probability Daenerys can win over Stannis's army by giving a speech is equal to:

    (1-50/100)*(1000/1600) = 0.3125 or 31.25%

    If the probability value is less than or equal to 0, then the main hero automatically loses the encounter. If the probability value is equal to 1 (or 100%), then the main hero automatically wins the encounter.

    b. If the main hero **wins**:
    - the losing hero retires, which means they are removed from the game. Your game program should print to the screen the message:
      `<Hero_name> has retired.`
    - the main hero acquires all of the losing hero's:
        - warriors (up to a maximum of 4),
            - if the number of warriors acquired by the main hero leads them to exceed the limit of 4 warriors, then the user should be presented with the option to pick which warrior(s) to keep, out of *all* of their warriors
        - *money*,
        - *influence*,
        - *army size*.
    - the locations controlled by the losing hero fall into the control of the winner

    c. If the main hero **loses**:
    - the encounter ends,
    - the main hero escapes,
    - but the main hero loses half their *influence* value

2. **Buy them outright:** the main hero relies on their *money* to buy out the other hero's army. If the *loyalty* of the first warrior in the opposing hero's party is low, the main hero can win the encounter. *Note*: If one hero were to control the entire map, and if they were the last hero left, they would have **17800** in *money*.

a. Compute the probability the main hero will win the encounter as:

$$\left(1 - \frac{opposing\ hero's\ 1st\ warrior\ loyalty}{100}\right) * \frac{main\ hero's\ money}{17800}$$

Example: Cersei has an *money* value of 8000. She is encountering Sansa Stark, whose hero Brienne has a *loyalty* value of 100. The probability Cersei can win over Sansa's army by buying it outright is equal to:

(1-100/100)*(8000/17800) = 0 or 0%

If the probability value is less than or equal to 0, then the main hero automatically loses the encounter. If the probability value is greater than or equal to 1 (or 100%), then the main hero automatically wins the encounter.

b. If the main hero **wins**:
- the losing hero retires, which means they are removed from the game. Your game program should print to the screen the message:
  `<Hero_name> has retired.`
- the main hero acquires all of the losing hero's:
  - warriors (up to a maximum of 4),
    - if the number of warriors acquired by the main hero leads them to exceed the limit of 4 warriors, then the user should be presented with the option to pick which warrior(s) to keep, out of *all* of their warriors
  - *money*,
  - *influence*,
  - *army size*.
- the locations controlled by the losing hero fall into the control of the winner

c. If the main hero **loses**:
- the encounter ends,
- the main hero escapes,
- but the main hero loses half their *money* value

3. **Battle:** both heroes rely on the *strength* of their warriors and the *size* of their army to win their encounter.

a. If either hero has no warrior but the other does, then the hero with no warrior automatically loses the battle.
b. If the main hero has more than one warrior, ask the user to choose which warrior they would like to represent them in battle.
c. if the opposing hero has more than one warrior, the game should choose at random one of their warriors to represent in battle (each with equal probability).
d. The hero with the highest value for
   (*army size) * (strength of chosen warrior)* wins the encounter.

Example: Jon Snow has an *army size* value of 20, and his warrior Jon Snow has a *strength* value of 100, for a product of 20*100 = 2000. He is encountering Stannis, with an army of 200, whose hero Melisandre has a *strength* value of 80, for a product of 200*80 = 16000. Thus, Stannis defeats Jon Snow.

You may resolve ties however you see fit.

e. If the main hero **wins**:
   - the losing hero retires, which means they are removed from the game. Your game program should print to the screen the message:
     `<Hero_name> has retired.`
   - the main hero acquires all of the losing hero's:
     - warriors (up to a maximum of 4),
       - if the number of warriors acquired by the main hero leads them to exceed the limit of 4 warriors, then the user should be presented with the option to pick which warrior(s) to keep, out of *all* of their warriors
     - *money*,
     - *influence*, and
     - *army size*.
   - the locations controlled by the losing hero fall into the control of the winner.

f. If the main hero **loses**:
   - **<u>the game ends</u>**!

"Valar Morghulis."

Note: If any hero is in proximity (see figure above) to any free roaming warriors, that hero will automatically acquire that warrior, without battle, as long as they don't already have 4 warriors in their party.

- If the hero already has 4 warriors in their party, the warrior continues to roam free until they encounter another hero.
- If two or more heroes are in proximity to the same warrior, no one will acquire the warrior that turn.
- If a warrior is acquired by a hero, *loyalty* and *morale* increase by 10 (not to exceed 100).

After the main hero (the user) has resolved all of their encounters, it's time to resolve possible encounters between other heroes. The same proximity rule applies: if one hero's location is within 2 tiles in any direction of another hero's location, the two heroes will have an encounter:

a. If both heroes have no warriors, then the hero with the largest army wins.
   - In the event of a tie, both heroes remain on the board at their respective locations, but lose half their army (rounding final army size down if the original army size was odd).

    - If both heroes have no warriors *and* no army, then both heroes retire.

b. If one of the heroes in the encounter has no warrior but the other does, then they automatically lose the battle.
c. If either of the heroes has more than one warrior, then choose at random one of their warriors to represent in battle.
d. The hero with the highest value for
   (*army size) * (strength of chosen warrior)* wins the encounter.

   Example 1: Euron Greyjoy encounters Sansa Stark. Euron has no warrior but Sansa does, so Sansa wins the encounter.

   Example 2: Euron Greyjoy encounters Daenerys Targaryen. Neither of them has a warrior, but Euron has a larger army size, so he wins the encounter.

   Example 3: Sansa Stark, represented by Brienne of Tarth encounters Jon Snow represented by Jon Snow. Sansa's battle value is 200*100. Jon's battle value is 20*100. So, Sansa wins the encounter.

e. You can decide how you would like to resolve ties.
f. If one of the heroes **wins**:
    - the losing hero retires, which means they are removed from the game. Your game program should print to the screen the message:
      `<Hero_name> has retired.`
    - the winning hero acquires all of the losing hero's:
        - warriors (up to a maximum of 4),
            - if the number of warriors acquired by the winning hero leads them to exceed the limit of 4 warriors, then the hero should keep all of their warriors from before the encounter, and then add, <u>in order</u>, as many of the losing hero's warriors, up to a maximum of 4.
        - *money*,
        - *influence*,
        - *army size*.
    - the locations controlled by the losing hero fall into the control of the winner.

"... A Mind Needs Books As A Sword Needs A Whetstone, If It Is To Keep Its Edge."

As the heroes move to a new location, they acquire more money and more knights, depending on the value of the new location.

- If the new location is a 'water' tile, the hero increases their *money* value by **10**
- If the new location is a 'land' tile, the hero increases their *money* value by **20**, and their army size by **10**.

- The cities provide different increases in *money* and *army size* **only** when the hero arrives **for the first time**, according to the values in the file `citiesGOT.txt.`

Note 1: If a hero already controls a location, they will ***not*** be able to acquire the money, knights and points by returning to that location over and over again.

Example:

- Cersei starts the game in King's Landing (the city represented by the character '`L`' on the map). She automatically gets control of this city by virtue of starting at that location.
- Suppose during the first turn, Cersei moves one position to the North.
- Suppose during the second turn, Cersei moves one position to the South, which would bring her back to King's Landing. If, at the beginning of this turn Cersei still has control of King's Landing (i.e., another hero might have moved through this city in her absence), then Cersei does ***not*** get the money, army size and points gains associated with controlling this city, because she already controls it.

Note 2: Update each hero's resources, only after you resolved all possible encounters.

Example:

- Euron Greyjoy starts the game in Pyke (the city represented by the character '`H`' on the map).
- Euron starts with 1000 in *money*, and 50 in *army size*.
- Let's assume that during the first turn, Euron moves one position to the North.
- Let's assume Euron has no encounters during this turn.
- He now occupies a 'land' tile, so his *money* increases by 20 to a total of 1020, and his *army size* increases by 10 to a total of 60.
- He would ***not*** regain these resources if another hero took control of that location, then Euron regained control over it

Note 3: After the resources have been updated, assign the heroes control over their new location.

Example:

- Euron Greyjoy starts the game in Pyke (the city represented by the character '`H`' on the map).
- Euron starts with 6 *points*, the value associated with the city Pyke.
- Let's assume that during the first turn, Euron moves one position to the North.
- Let's assume Euron has no encounters during this turn.
- He now controls a 'land' tile, so his *points* increase by 2, to a total of 8 *points*.

Note 4: If one location was previously controlled by another hero, change the owner of that location to the new hero. The new hero will acquire the money, knights and points bonuses from that location. The previous owner loses the *points* for the location, but keeps any *money* and knights (*army size*) previously acquired.

Example:

- Cersei starts the game in King's Landing (the city represented by the character '`L`' on the map). She automatically gets control of this city by virtue of starting at that location.
- Let's assume that at some point during the game, Cersei is ***not*** in King's Landing. At this point, let's assume that Cersei has 8100 in *money*, 350 in *army size*, and 58 *points.*
- Let's assume that another hero, Sansa moves onto the King's Landing location. Before moving, Sansa had 2000 in *money*, 150 in *army size*, and 36 *points*.
- As Sansa moves into King's Landing, she becomes the "owner" of that city, and Cersei loses control of it.
- The resources and points values update as follows:
    - Cersei:
        - *money* does not change
        - *army size* does not change
        - loses the *points* for King's Landing (35) as is now at 58 - 35 = 23 *points*
    - Sansa:
        - acquires the *money* for King's Landing (175) and will now have:
          2000 + 175 = 2175
        - acquires the *army size* for King's Landing (175) and will now have:
          150 + 175 = 325
        - acquires control over the city and gains the *points* (35). She will now have:
          36 + 35 = 71 *points*

After resources and control have been updated, the following events can occur:

1. If the user has arrived for the first time in a new city, inform them that they have arrived in a new city and ask them if they want to listen to rumours:

   ```
   You have arrived in King's Landing. Do you want to listen to
   rumours? Y/N: Y
   ```

   If the answer "Yes", reveal the distance in number of moves (or "days of travel") to the closest hero.

Example:

- Suppose Sansa Stark's location is in Winterfell (row 8, column 5).
- Suppose Jon Snow's location is in Eastwatch by the Sea (row 5, column 7).
- The distance between the 2 heroes is 5 (i.e. it would take one of them 5 moves/turns to reach the other).

$$distance = |row1 - row2| + |column1 - column2|$$

2. If the user has arrived for the first time in a new city, inform them that they have arrived in a new city and ask the user if they want to buy more knights:

   ```
   You have arrived in Myr. Do you want to buy more knights? Y/N: Y
   ```

   If the answer “Yes”, the user can spend money to increase army size. The price of one knight is 20 units of *money*. This is in addition to the army size increase from regular recruitment in the city during this turn when the user’s hero arrives, and this opportunity only arises from the excitement of a new city ruler.

3. If the user landed on a location which has *dragonglass*, then ask the user to choose one warrior to be equipped with *dragonglass*.

   ```
   Great news! You have stumbled upon Dragonglass. Choose which one
   of your warriors would you like to equip with Dragonglass (enter
   their name or number): 1
   ```

   Note: you can choose to implement this part however you wish to. You can present the user with menu options, ask them to supply a full name or a number. It’s up to you.

4. If any other hero landed on a location which has *dragonglass*: choose one warrior at random from their party to be equipped with *dragonglass*.

5. If a free roaming warrior landed on a location which has *dragonglass* they will be equipped with *dragonglass*.

6. If the user or the other heroes land on a location with *dragonglass* but they do not have any warrior in their party, they cannot acquire the *dragonglass*. That location will continue to have *dragonglass* until another hero or warrior arrives at that location.

7. If any party acquires the *dragonglass*, the location will not have *dragonglass* anymore, and it will become a plain land tile (‘p’).

“Winter Is Coming.”

At the end of every turn, there is a 30% probability that one random event can occur. If an event occurs, then it is one of the following events, each with equal probability:

1. **Drought**: there has been no rain in days and the army has very little food left.
   - *morale* and *strength* of each of the warriors associated with the main hero (the user) decrease by 10.

2. **Deserters**: this campaign is taking a very long time and the soldiers are tired and sad. Some decide to leave.
    - *army size* for the main hero (the user) goes down by 10.
    - *loyalty* for each the warriors associated with the main hero (the user) goes down by 10.

3. **Jackpot**: your heroic cause has won the hearts of many noble and wealthy benefactors! They generously donate funds to your cause.
    - the main hero's *money* value will double

4. **Kindness**: you decide to offer shelter to some refugees.
    - the *army size* for the main hero (the user) goes up by 10.
    - the *loyalty* of each the warriors associated with the main hero (the user) increases by 10.

Note that the *loyalty* and *morale* values for warriors cannot exceed 100, and *strength* cannot exceed 200, and none of these can become negative. The *army size* for the main hero cannot become a negative value.

Some events are preconditioned to occur at set times:

1. `Daenerys` acquires `Dragons` as their warrior after 10 turns.
    - `Dragons` are special, mythical creatures:
        - their *strength* value is 180
        - their *loyalty* value is 100
        - their *morale* value is 80
    - `Daenerys'` *army size* will also increase by 5000
    - If `Daenerys` already has 4 warriors, then `Dragons` eat the last warrior to join `Daenerys`. The devoured warrior is removed from the game and the *strength*, *loyalty* and *morale* of `Dragons` all increase by 10% relative to the numbers above, thanks to the tasty meal.

2. `Stannis` gets consumed by his demons and dies after 10 turns.
    - `Melisandre` and `Davos Seaworth` (and any other warriors attached to Stannis) become free roaming warriors.
    - Your game program should print to the screen the message:
      `Stannis has retired. Press F to pay respects.`
        - Then, if the user enters `'f'` or `'F'`, print to the screen: `Mood.` and continue the game.
        - But if the user enters anything besides `'f'`/`'F'`, ask the user: `"Have you no respect?! Press F to pay respects."`

3. `Sansa` acquires warrior `Arya` after 10 turns, regardless if `Arya` is free or attached to another warrior at that time.

4. `Cersei` loses warrior `Jaime` after 10 turns.

    ○ `Jaime` becomes a free roaming warrior.

"Death Is So Terribly Final, While Life Is Full Of Possibilities."

The game ends when:

1. The user loses a battle with another hero. In this case, the user loses the game.

> Express your disappointment for the defeat and offer the user the choice to play again, from the beginning of the game.

2. The user fights the final battle with the *White Walkers.* If the user wins the final battle, then they win the game. If the user loses the final battle, then they lose the game.

> Any of the following conditions will trigger the final battle with the *White Walkers*.

- After 50 turns (you decide if it is in turn 50 or 51), or
- If the *army size* of the user meets or exceeds 2000, or
- If the user controls at least 50% of the board (200 tiles), or
- If the user has a score of 400 points, or
- If the user is the only hero left.

Fig 2. White Walkers *want to turn* **you** *into a yeti!*

Once at least one of these conditions are met, each turn, the following message alerts the user that the *White Walkers* are coming to attack:

```
Sad news! The White Walkers are approaching our location. Would
you like to battle them now? (Y/N): N
```

The user has up to 5 turns to accept the battle with the *White Walkers*. The user can deny the battle 4 times (by entering **`N`**), and use this time to try to find *Dragonglass*, if they have not yet acquired it. If the user refuses to face the White Walkers for 5 turns, then they lose the game.

If the user accepts the battle with the *White Walkers*:

- A *power score* for the hero's party is calculated, based on characteristics of the hero and all warriors attached to that hero.
  - A hero's power score starts out equal to their army size
  - Each warrior increases the hero's power score by: $(warrior's\ strength) * \frac{(warrior's\ morale)}{50}$
  - A hero's influence and empire also strengthen their army's resolve to defeat these yeti-loving, icicle-toting thugs. Their power score is further increased by the amount: $(\#\ cities\ the\ hero\ controls + 1) * \frac{(hero's\ influence)}{4}$
  - If a hero has *Dragonglass*, then their power score increases further by 10000
- The battle with the *White Walkers* consists of 3 rounds of combat. To win the battle, the hero must win at least one of these rounds of combat.
- The probability that the hero wins any given round in this final battle is given by:

  $1 - e^{-(power\ score)/30000}$, where *e* is the exponential constant

"The North Remembers."

For each game won, before starting a new game (or before ending the program), record the user performance (**number of points**) in a file named **resultsGOT.txt**. If there are previous users recorded in the file, the current user's performance should be appended at the end of the file. The output file should have *tabulated values* in the following format:

```
<Name>         <Hero>         <points>
George Martin  Sansa Stark    525
Robb Stark     Jon Snow       478
```

If the user has not chosen one of the default 6 heroes, the value for the hero's name should be `Self`. Note the *gorgeous* alignment of the names, heroes and points. Ahhh… so satisfying!

Very important:

Think of the game you are developing as a real application. Try to anticipate user error and respond accordingly. For example, if the user can only choose between 4 menu options but they choose option 5, your game should not crash, but instead display appropriate error messages and ask the user to choose again. Other examples of user error: trying to explore locations outside the map size or entering a negative army size. In general, when testing your program, it is recommended that you try to “break” it. Fix all the instances that would cause errors or unwanted game outcomes, so that during the grading interview your program is unbreakable.

Extra credit (10 points total possible):

#1: (3 points) At the end of each game, present the user a sorted list of the top 5 players of the game and their stats: name, hero, number of points.

#2: (3 points) Introduce a new piece of “equipment” besides ships. In your PDF report, document your new variable and how it influences the rules and outcome of the game. You must significantly alter the flow of the game for your change to be awarded these points.

#3: (4 points) present your project in a video, or in class (see below)

Minimum Requirements:

Your implementation of the Game of Thrones should have:

- At least 4 user-defined classes
- At least 2 of these classes should have 4 or more data members.
- At least 1 class must include an array of objects from a class that you created.
- Appropriate methods for each class (including getters and setters)
- Your project implementation must include at least:
    - At least 6 **if** / **if-else** statements
    - At least 4 **while** loops
    - At least 4 **for** loops
    - At least 2 nested loops (these count in the 4 **while** and 4 **for** loops above)
    - At least 7 strings variables/data members
    - Reading from files (the provided text files or files of your own creation)
    - Writing to a file (`resultsGOT.txt`)
- Your project must have an interactive component (ask the player for input, create menus for choices, and so on). In other words, it’s a *game*!

Project timeline:

Checkpoint 1: Choose Project, meet with TA or CA before Wednesday, April 10 @ 6 PM

If you choose to implement the Game of Thrones project, you will need to come up with your own program structure. You will need to decide how the information will be stored in objects and how it will be passed between objects. You need to choose which classes to define, what are their data members, and which class/object is responsible for each part (functionality) of the game.

The choice of classes is entirely up to you. But to ensure you did not choose an approach that is not feasible, we require you meet for 15 minutes with the professor, a TA, or a CA, to go over your classes. **This meeting is mandatory**. You must choose a slot from the Moodle scheduler and have the meeting before Wednesday, April 10 at 6 PM. This meeting will count as interview grading in terms of the no-show and (re)scheduling policies. See the **syllabus** for more information.

Note: Even if you choose to create your own project, the design meeting is mandatory. Follow the instructions for the Project Proposal in the *ChooseYourOwn* Project write-up.

Checkpoint 2: Submit class files & Code Skeleton via Moodle, due Wednesday April 10 @ 11 PM

Your .h files should be *complete* with all the data members and member functions you will be using for each class. For the class implementation .cpp files, you should fully implement simple functions like your getters and setters. For more complex functions you can include function stubs with detailed comments. At a minimum, the input parameters, output type and pseudocode description should be present.

For example, if we were stubbing a function to implement bubble sort and return the number of swaps we might give in our code skeleton:

```
/*
   1. Compare adjacent elements. If the first is greater than the
      second, swap them.
   2. Do this for each pair of adjacent elements, starting with the
      first two and ending with the last two. At this point the last
      element should be the greatest.
   3. Repeat the steps for all elements except the last one.
   4. Repeat again from the beginning for one less element each time,
      until there are no more pairs to compare.
*/
int bubble_sort(int arr[], int size)
{
      int swaps = 5;
      return swaps;  // function returns expected type (int)

}
```

Your Code Skeleton should be inside your driver file and it should contain detailed comments with pseudocode explaining the functionality of the project.

You must submit your Class Files and Code Skeleton to Moodle to get full credit for the assignment. Failure to submit the Class Files and Code Skeleton will result in a 10-point penalty on your Project score.

Checkpoint 3: Final Deliverables due Wednesday April 24 @ 11 PM

The final version of your project will be due on **Wednesday April 24 @ 11 PM** (no early submission bonus available and there will be no extensions). You must submit a .zip file to Moodle which includes:

- All .h and .cpp files, including the main driver program, correctly indented and commented.

Checkpoint 3A: Project Report - Reflection Activity, due Sunday April 28 @ 11 PM

Write a 1-2 page report containing answers to the following **Reflection questions:**

1. How did you prepare for the Project?
2. How did you develop your Code Skeleton? In what way(s) did you use your Code Skeleton?
3. Reflect on how you could have done better, or how you could have completed the project faster or more efficiently.
4. In addition, write a paragraph answering the following question, in the context of the Project in CSCI 1300:

   Did you have any false starts, or begin down a path only to have to turn back when figuring out the strategy/algorithm for your Final Project program? Describe in detail what happened, for example, what specific decision led you to the false starts, or, if not, why do you think your work had progressed so smoothly. In either case, give a specific example.

<u>Note: all reflection papers must be individual.</u>

Submit a PDF file of your report to the Moodle dropbox by **Sunday April 28 @ 11 PM**

Failure to submit the Report will result in a 10-point penalty on your Project score.

Checkpoint 4: Interview Grading - will begin on **Thursday April 25**, and must be completed at the latest on **Friday May 3** (reading day).

Checkpoint 5: Extra Credit Opportunity #3- 4 points (also for students choosing GOT) (due Sunday, April 28th @ 11 PM)

- Make a 5 minute (+ or - 1 min) video explaining:
  - The project idea
  - Implementation and approach
  - A demonstration of the working project
  - A **Google form** will be made available to **submit a link to your video**.
- **OR** you can present your project in lecture (5-7 min presentation).

        - You must **sign up to present by Wednesday, April 24 @ 11:55 PM**
        - Limited slots will be open and will be first come, first serve
        - Presentation should include:
            - project idea,
            - implementation and approach, and
            - a demonstration of the working project.
    - You **cannot** earn extra credit for doing both the video and the presentation in class

Collaboration:

All work for this assignment (and course in general) must be your own, original work. You may work together, but your assignment submission is **your own**. Your work may not include code taken from online resources like Chegg or StackExchange, or from other students (past or present), even with modification. Any such instances constitute Academic Dishonesty (passing off others' work as your own) and will earn you a 0 on the assignment and a trip to the Honor Code Council. If you aren't sure if something is okay, then please just ask!

Points:

Note: <u>If your code does not compile, you can get a maximum of 40 points for the project.</u>

20 points - meets minimum requirements specified on the first page (# of classes, loops, etc.)

40 points - interview grading
- TA's questions about your project
- algorithms descriptions, comments, good style
- code compiles
- if your code does NOT compile, you can get at most 20 points from meeting the minimum requirements, and at most 20 points from the interview

40 points - project functionality
- the game plays at outlined in the project description
- your solution accounts for user error

3-10 points extra credit
- (3 pts each) Extra features you can build into your game program
- (4 pts) Create a 5 min (±1 min) video explaining your project implementation & demo, ***or***
- sign-up to present and demonstrate your project in lecture (5-7 min presentation).

Possible Deductions:
- 10 points for not attending the project meeting (Checkpoint 1)
- 10 points for not submitting code skeleton and class files (Checkpoint 2)
- 10 points for not submitting report (Checkpoint 3A)