Assignment 05
https://docs.google.com/document/d/1kMYwNoQwc9P_Jfz3sf3CODth-UVsAV3iDRfcxDIOEXA/edit?usp=sharing

Due: Wednesday, March 16th, 2022, at midnight, 100 points.

Documentation Header Reminder

Before you start your assignment, you will need to add documentation similar to what we demonstrated in the first few lectures.

Function Prototype Documentation Reminder

In the function prototype section of your C++ program, remember to add documentation to each function similar to what was demonstrated in lectures.

Separate File Compilation

For this assignment (and onward), you will submit **multiple** C++ compilable files containing a program written in C++. Name your file a meaningful name and give it a proper extension (.h, .hpp, .cpp). Also, make sure that you compile and run your program using the GNU (g++) compiler before submitting to make sure that it will work.

HOME DEFENDER

Background

Congratulations, you recently became a new homeowner, and you also got a good deal for how much you paid for the acres of land of your new house! Sadly, after spending several weeks of sweet moments living in the new house, you soon realize that some of your surrounding neighbors are jealous of the great deal you got on the house, and some of them decided to commit bullylike acts by trying to steal some of your land. Fortunately, some of the neighbors are good, and occasionally, they will offer some much needed help to you. Yes, you have thought about selling off the house, but after consulting a Magic 8-ball, you decided that you'll defend your new home with all you got.

Specifications

In your program, you will create different "structs" to complete the program.

- The **NewHomeowner** struct
  - Data members

        - The new homeowner should have a name and the address of the new home
        - They should keep track of the total acres of the land the new house sits on for at the start and at the end of the program
        - A variable to represent the total number of garages
- The **Neighbor** struct
    - Data members
        - A neighbor should have an ID
        - They also keep track of the total acres of the land their house sits on for at the start and at the end of the program
        - The total number of garages
        - A variable to indicate whether they are a good neighbor or a bad neighbor
        - A variable to indicate whether they are *active* or *inactive* in the program
            - A good neighbor should always be active

Besides the required structs, the following functions must also be properly implemented. You may add more or use different parameters if you like, unless specifically stated otherwise. Variable names may be different in your code. Return types and parameter types are up to your own discretion.

initializeNeighbor()

- The main purpose of this function is to initialize the attributes of a neighbor with random values.
    - Each neighbor in the program should be given a *unique* identification number, starting out with **0**.
    - The original acres of land should be a random number chosen between **19** and **50**, inclusive. As for the value of the final acres of land, it doesn't matter much in this function; simply make sure that it is updated properly throughout the program and at the end.

- The number of garages for a neighbor should be set as the *tenth digit* of the original acres of land, i.e. *numGarages = 2* if *originalAcres = 27.*
- To determine whether a neighbor is good or bad, it should be based on a *50/50* probability model.

requestAssistance(theNewHomeowner, aGoodNeighbor)

- This function should be called when the new homeowner encounters a good neighbor. If the new homeowner currently has fewer garages than the good neighbor, then there will be a **80**% chance that the good neighbor will donate a garage to the homeowner. On the other hand, if the number of garages owned is more than (or equal to) the good neighbor, then the chance drops to **30**%.

defendHomeland(theNewHomeowner, aBadNeighbor)

- This function should be called when the new homeowner encounters a bad neighbor. If the new homeowner currently has fewer garages than the bad neighbor, then there will be a **53**% chance that the bad neighbor will successfully steal some acres of land from the new homeowner.
  - The amount of land the bad neighbor will take is going to be the *difference* of the number of garages owned by the two parties.
  - After stealing the land with bad conscience, the bad neighbor also has the opportunity to build a new garage with the new resources. This chance is going to be **46**%.
- On the other hand, if the number of garages owned is more than (or equal to) the bad neighbor, then the operation of home defense is a successful one. Furthermore, if the homeowner managed to get an *even* random number between **1** and **80**, inclusive, then there's a **73**% chance that *half* of the stolen land from *this* bad neighbor could be reclaimed, and a **27**% chance that *all* of the stolen land from this bad neighbor could be reclaimed. If this bad neighbor has not managed to steal any land before, then this procedure can be skipped.
  - A bad neighbor becomes *inactive* when all stolen land is reclaimed.

Overall Program Flow

1. Congratulate the new homeowner and initialize their info.
2. Generate **6** neighbors and randomly populate their info.
   a. These neighbors should be stored in a C++ *array*.
   b. If there are NO bad neighbors generated, then end the program since everyone is happy.

3. If there are bad neighbors' presence, then the new homeowner needs to begin defending the home, and hopefully can put a stop to the bullies once and for all. The program will continue by iterating through the neighbor array. For each encounter with a good or a bad (must be active) neighbor, call the respective function. Each iteration counts as one round.
4. The program should end when the new homeowner has no acres of land left, or when all the bad neighbors become inactive. The program should also terminate after a maximum of **10** rounds if the two stopping conditions were not not met.

Notes

- Variables representing acres of land should be *floats*.
- Seed the random number generator with **327** (e.g. *srand(327)*).
- You may define additional functions to be used to improve the program's organization.
- You may use functions from existing libraries if 1) they are explicitly stated in the assignment or 2) the usage of the library functions has been thoroughly introduced in class. Otherwise, the usage of functions from existing libraries is prohibited without the permission of your instructor.
- If the parameters and return type of a function is not specified, then it is your responsibility to determine the most appropriate function signature.

Sample Outputs

Coming soon…