Assignment 09 (Also the Final Programming Project)

Due: Tuesday, May 3rd at 11:59 PM 100 points.

- ★ This assignment usually is counted toward 10% of the course grade, check the course syllabus for your corresponding section.
- ★ You are allowed to work with at most one other person on this final project who needs to be in the same section as you are. Make sure to put both of your names in the header of all the C++ files. Only one of you needs to submit to GitLab.

Link:
https://docs.google.com/document/d/1FFL24H78o7jAtwkSwS0kg2dVn7hmUSlF4RpbNfsFVQQ/edit?usp=sharing

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

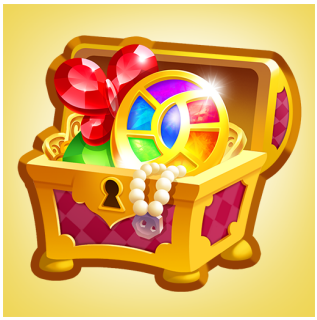
For this assignment (and onward), you will submit multiple C++ compilable files containing a program written in C++. Name your file a meaningful name and give it a proper extension (.h, .hpp, .cpp). Also, make sure that you compile and run your program using the GNU (g++) compiler before submitting to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

Background

While you are working hard to try to get as many dates as possible in the big city, a group of skillful robbers is also working hard, hoping to add more funds to their retirement accounts by deciding to rob a rich bank together[1]. However, during their attempt, an incident occurred and caused all of the valuable jewels in the bank to end up scattering all over the city streets. The robbers are now rushing to recollect their prize, but of course the bank alarms sounded and cops are now just showing up too. You know that an epic chase is about to happen in the city grid, and you just couldn't resist watching it through your apartment window even though your love interest has been calling your mobile more than ten times now…

Specifications

- The City Class
  - This class should have the following attribute(s):
    - A 2D array of type character (**char**) with size 10 x 10.
    - A current count of the number of jewels scattered in the city grid
      - The number of jewels in the city grid is subject to be changed as police and robbers move during the chase.
- The Jewel Class
  - This class should have the following attributes:
    - The **value** (or the worth) of the jewel
    - The **coordinate** that it was originally scattered at in the city grid



- The Robber Class
  - This class should be created as a template class with just one template parameter. The parameter controls the type of items that a robber is interested in. For this assignment, it's going to be the Jewel type, but in the future, we could easily change that to be another type, i.e. Stuffed_Animal.
    - It is safe to assume that the template parameter type will always have the *value* and *coordinate* member attributes.
  - A robber should have the following attributes:

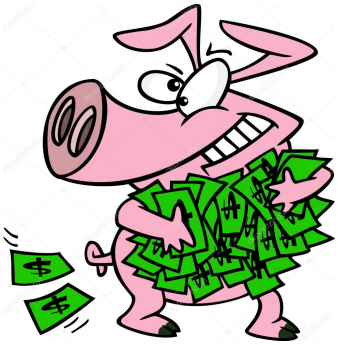
- An unique **id**
- The current **coordinate** for the robber's position in the city grid
- A **bag** to hold the loot (i.e. jewels)
    - The maximum capacity for the bag should be set to 17.
- Another current **count** of the total worth of the loot collected by ALL the robbers so far. This should be a static member variable.
- A variable indicating whether the robber is still **active** (inactive corresponds to being arrested by the cop)
- A variable representing the type of the robber: *ordinary* or *greedy.* Each type of a robber *moves* differently in the program.

- A robber should have the following function(s):
    - The ***pickUpLoot()*** function
        - Simply insert the loot into the robber's bag if it is not full. If the bag is already full, then perform nothing.
    - The ***move()*** function
        - This function should be used to move the robber one step in a specific direction. The direction should be determined by a random number between 0 and 7.

0 (NW)	1 (N)	2 (NE)
3 (W)	Robber	4 (E)
5 (SW)	6 (S)	7 (SE)

            - Only **valid** movements are allowed. No one is allowed to move outside of the city grid border.
            - [**Bonus** (15 points)] This bonus implementation only applies to the greedy robber type. When getting the random direction, the resulting direction needs to be guaranteed that it contains at least one jewel in its path (to the border of the city grid). If such a direction does not exist (i.e. the current jewel count in the city is zero) , then simply move in a random direction. Make sure to state that you are doing this bonus in your files so the graders won't miss it.
        - Possible scenarios after moving to the new spot:
            - You ran into your friend, another robber:

        - Nothing bad would happen. You happily greet your friend, cheerio!
        - However, if you are a greedy robber, when you bump into another robber, your overexcitement will cause *half* of your already collected loot to fall out of the bag! You'll need to redistribute the fallen loot back to their original coordinates in the city grid. If the grid is already occupied by any entity, then choose another random location to place it at.
            - Only the **moving** greedy robber will lose half of the jewels, even if the robber being bumped into is another greedy robber.
            - The suggestion is to **round down** if the number of loot in the bag turns out to be an odd number.
    - You found a loot at the new spot:
        - Put it into the bag by using the *pickUpLoot()* function. You should be properly recording its coordinate and estimated value:
            - The estimated value of the loot should be computed as the summation of the grid coordinate. For instance, the value for the loot would be $14 if it's found at coordinate (7, 7) in the city grid.
        - Moreover, if you are a greedy robber, you'll need to check this: if the estimated value of the loot turns out to be an *even* value, then the greedy robber gets to move again!
            - The greedy robber can only move 3 times consecutively though.
    - You ran into a cop, a police:
        - Too bad, you get caught by the police and the *arrest()* function from the Police class will be called. Your status should then be set to inactive and can no longer participate in the chase.



- The Police Class
    - This class should have the following attributes:
        - An unique police **id**
        - The current **coordinate** for the police's position in the city grid
        - A current **count** of the total worth of the loot confiscated so far
        - The total number of robbers caught by the police
    - A police officer should have the following function(s):
        - The ***arrest()*** function
            - This function is invoked whenever a robber gets caught by the police. During the arrest, the police should diligently record the worth of the confiscated loot and update the number of robbers detained by incrementing one.
        - The ***move()*** function
            - This function should be used to move the police one step in a specific direction, identical to that of an ordinary robber.
            - Possible scenarios after moving to the new spot:
                - You ran into a robber:
                    - Arrest the robber immediately!
                    - If there is a group of robbers at the spot, then the police should arrest all of them.
                - You found a jewel at the new spot:
                    - Return it to the bank and remove it from the city grid.
                    - Remove it from the city grid because you're for sure going to return it to the bank after the chase is over.

Overall Program Flow

1. The simulation of the chase should begin by first creating the city grid.
2. I think you are a better approach is to create to create one great
3. Randomly scatter 47 jewels in the city grid by notating them with 'J' in the grid.
    a. Each cell in the grid can be occupied by only one jewel.
4. Create one police, two ordinary robbers, and two greedy robbers. Randomize their initial locations in the grid and notate them with 'p' and 'r' correspondingly.
    a. Each cell in the grid can be occupied by only one entity in the beginning.
5. Print out the initial state of the city grid nicely (you might want to create a function to handle this specifically).

6. Let the chase begin! Each turn consists of having the ordinary robbers move first, followed by the greedy robbers, and then the police.
    a. Remember, a robber stops moving once they get caught. If a robber gets arrested, then the remaining robbers can still continue the chase.
7. At the end of each turn, print out the state of the city grid nicely.
    a. If multiple robbers occupy the same cell, then notate the cell value with 'R'.
8. We will do a maximum of 30 turns. If the police fail to catch all the robbers in the final chase, then all the robbers get to run away freely, but the confiscated loot stays with the police.
    a. The robbers could also win if they managed to collectively pick up enough jewels with a net worth of $438 at any point during the chase.
        i. If this happens, then the police will release all the arrested robbers as well since the bribe is too tempting. Also, the police get to keep all the confiscated loot too.
9. When the chase terminates, print out a summary of the chase outcome. For example, the format should be:

```
Summary of the chase:
    The robbers wins the chase because maximum turns (30) have been reached
    Police id: 1
        Confiscated jewels amount: $512
        Final number of robbers caught: 0
    Ordinary Robber id: 1
        Final number of jewels picked up: 5
        Total jewel worth: $91
    Ordinary Robber id: 2
        Final number of jewels picked up: 0
        Total jewel worth: $0
    Greedy Robber id: 3
        Final number of jewels picked up: 22
        Total jewel worth: $75
    Greedy Robber id: 4
        Final number of jewels picked up: 8
        Total jewel worth: $240
```

Additional Notes

1. As a reminder, trying to get rich by robbing a bank is ALWAYS a bad idea. It is MUCH safer to just get a job as a programmer.
2. Set the random seed to be 85 for this assignment.
3. It is recommended that each class definition goes in their own header file.

4. For all the data member variables and functions in the class definitions, it is up to you to determine their proper access levels (i.e. public vs private).
5. It is also your sole responsibility to determine the best return types and parameters for the functions specified in this assignment.
6. You may create additional member attributes in the classes if proven to be helpful.
7. You may define additional functions to be used to improve the program's organization. This includes global functions and member functions for the classes as well.
8. 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.