CSCI 1300 CS1: Starting Computing
Naidu/Correll/Yeh/Hoefer - Fall 2021
Project 3: Where in the Code is Carmen Sandiego

<u>WHERE IN THE CODE IS CARMEN SANDIEGO</u>

For the Final Project, you will implement a text-based adventure game in C++. This game adds a creative, qUiRkY spin to the classic video game “Where in the World is Carmen Sandiego?” In general, your mission (should you choose to accept it), you mega-elite haxxor lord, is to defeat 8 of carmensandiego101’s hackers to try and stop her nefarious information-stealing! Overall dynamics consist of navigating across server rooms, weeding out hackers, and engaging in combat with them. Along the way, you’ll acquire new languages and debugging skills and learn how to lower your frustration on StackOverflow to prevent rage-quitting.

The game can be summarized by these core concepts:

1. The player buys supplies from Best Buy before starting the journey through the hacker-infested server rooms.
2. There are five server rooms that you must explore in order to find hackers. Once you’ve defeated between 1 - 3 hackers in the current room, you can move onto the next server room.
3. Along the way, there are opportunities to i) visit Best Buy to buy items and upgrade your equipment, ii) interact with potentially friendly non playable characters (NPC), iii) fight hackers that you find during your exploration of the server room.
4. The game ends when the player has defeated, between 1 - 3 hackers in each of the 5 server rooms, when Carmen and her hackers progress to 100 steal all of the server’s top-secret files, when the player’s computer maintenance level reaches 0, or when the hackers succeed in getting the player so frustrated that they rage quit.

Initial Conditions:

- At the beginning of the game, the player is asked to enter their name.
- The player is then placed in a Best Buy store. Before starting, the player can purchase supplies from Best Buy. Best Buy has the following supplies for sale: computer parts, antivirus software, virtual private network, and internet provider. More details on Best Buy in the next section.
- After the player is done shopping, they are placed into the first server room.
- The player starts with a frustration level of 0. The game ends if the player's frustration level reaches 100.
- The player starts with 200 Dogecoins.
- The player has one computer.
- The player starts with internet provider level 1.
- The player starts with one virtual private network (VPN).

Visit Best Buy:

At the beginning of the game, the player has the opportunity to purchase things they might need from Best Buy. Initially, the player is presented with information about what they might need from Best Buy and why. In case the user wants to level up when inside a server room, there is also 1 Best Buy located in each server room at a random location (which you are in charge of spawning). Below is an example excerpt for the game, which you can also find in the file *"best_buy_info.txt"*.

```
You have 200 dogecoins, 1 computer and 1 VPN. You will
need to spend the rest of your money on the following
items:

- COMPUTER PARTS. If your computer breaks, you need 5
computer parts to make a new one.
- ANTIVIRUS SOFTWARE. If your computer is infected with a
virus, use the antivirus software to get rid of it.
- VIRTUAL PRIVATE NETWORK (VPN). The more VPNs you have
the harder it is for a hacker to infect your computer with
a virus.
- INTERNET PROVIDER. The better the internet provider, the
more reliable your hacking will be.

You can spend all of your money here before you start your
journey, or you can save some to spend on a different
electronics site along the way. But beware, some of the
```

```
websites online are shady, and they won’t always give you a
fair price...
```

Present the player with the information about possible purchases. You can use some of the text in the fragment above or you can modify it to include more information.

Then, ask the player if they wish to purchase things in the following 4 categories:

1. Computer Parts:

a. Best Buy recommends that the player should purchase at least 1 of each computer part in case your main computer breaks. You can have a maximum of 3 of each computer part and 1 extra premade computer. These are the different parts:
    - A CPU costs 10 Dogecoins
    - A GPU costs 20 Dogecoins
    - A Power Supply Unit costs 5 Dogecoins
    - A Computer Case costs 15 Dogecoins
    - An internet card costs 5 Dogecoins
    - A keyboard and mouse costs 10 Dogecoins
    - A premade computer costs 100 Dogecoins

b. Ask the player how many computer parts they wish to purchase.

c. Compute and print the current total cost so far.

2. Antivirus software:

a. One USB stick of antivirus software costs 10 Dogecoins and will give you a one-time use to get rid of any viruses on your computer.

b. Ask the player how many antivirus USB sticks they would like to purchase.

c. Compute and print the current total cost so far.

3. Virtual Private Network (VPN):

a. A VPN costs 20 Dogecoins each. VPNs reduce your chances of getting hacked by preventing hackers from tracking down your main computer. The increase in security maxes out at 2 VPNs.

b. Ask the player how many VPNs they would like to purchase.

c. Compute and print the current total cost so far.

4. Internet Provider:

a. Internet provider level 2 costs 10 Dogecoins

b. Internet provider level 3 costs 25 Dogecoins

c. Internet provider level 4 costs 40 Dogecoins

d. Internet provider level 5 costs 50 Dogecoins
e. Ask the player what internet provider level they would like to buy. They can upgrade their internet provider at a later time if they so desire.
f. After the player selects an internet provider, compute and print the current total cost so far.
g. Having a better internet provider increases the player's chances of winning a hacker battle.

Note 1: At any time in the buying process, if the player is attempting to buy an item (or a number of items of the same kind) whose price exceeds the money the player has in hand, print a message to the player informing them they do not have enough money and ask them to choose another quantity of the item in question.

Note 2: The player can choose to purchase items in a certain category multiple times during the same store visit. For example, the player might choose the purchase of VPNs, followed by computer parts, then VPNs again, then more computer parts, then an internet provider, but no antivirus. After every new set of items added, your program should print the total cost so far.

Best Buy will be located at a random position in each server room except for the first room where it will be located at the player's initial position. The prices at the stores along the trail get progressively higher as you get further along. In the second server room, the prices will be **10% higher** than the ones at the starting location. In the third room, they will be **20% higher**. In the fourth room, **25% higher**. In the fifth room **30% higher.**

During your mission, after each move, you will automatically mine a certain amount of Dogecoin that will be added to your Dogecoin amount. No matter what, **you will gain 5 Dogecoin each turn**. With each GPU that you have in your inventory, you can gain an **additional 5 Dogecoin.** For example, if you have 2 GPUs, you will be able to get 15 Dogecoin after each move.

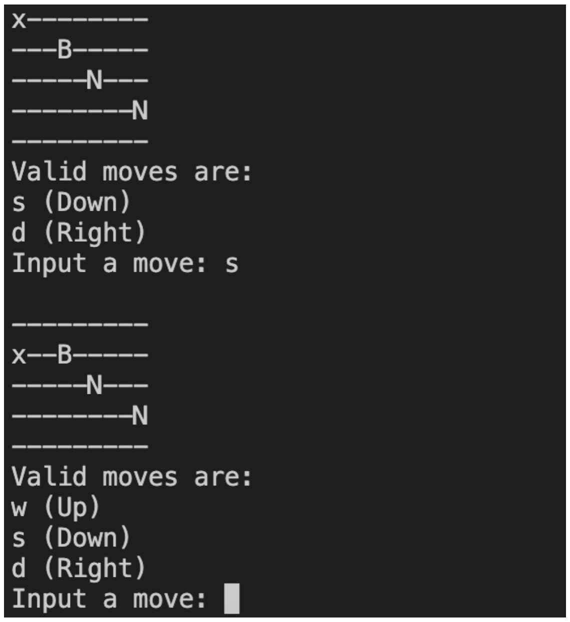
Using the given Map class:

We provide a fully-implemented Map class that you will need to figure out how to use in your code. The Map class defines a 2D map of size 5x9 and stores data in a mapData array of type char. This array contains characters that represent player, Best Buy, NPC, and Hacker locations as well as empty spaces. Setters and getters are provided for player position. Spawn functions are provided for Best Buy, NPC, and Hacker that places these entities in mapData. Functions for displaying the map in the terminal and moving the player around the map are also given.

mapDriver.cpp shows how you may use the class by first initializing a Map object and spawning any entities relevant to this game (NPC, Best Buy, Hacker). The for-loop allows the player to execute 10 moves around the map, displaying the map after each move and printing a confirmation message if the player lands on any of the spawned entities.

You may modify this class in any manner you see fit -- by changing the types of entities, the size of the map, etc. But you must use a 2D map as part of your project.

'x' corresponds to the player position, 'B' represents the Best Buy location, and 'N' is the NPC location. Hacker locations are not shown on the map. Player always starts at (0, 0). An example output will look something like this:



```
x---------
---B------
-----N----
---------N
----------
Valid moves are:
s (Down)
d (Right)
Input a move: s

----------
x--B------
-----N----
---------N
----------
Valid moves are:
w (Up)
s (Down)
d (Right)
Input a move: 
```

The map class will have seven functions that you can use.

1. void displayMap()
    a. Prints the map on the terminal as seen above.
2. void displayMoves()
    a. Prints the valid moves a user can take using the w, a, s and d keys
3. int getPlayerRowPosition()
    a. Returns the players row position index
4. int getPlayerColPosition()
    a. Returns the players column position index
5. bool isNPCLocation()
    a. Returns true if the player's current position is the NPC location
6. bool isBestBuyLocation()

    a. Returns true if the player’s current position is a Best Buy location
7. bool isHackerLocation()
    a. Returns true if there is a Hacker at the player’s current position
8. bool executeMove(char move)
    a. Takes the user’s next move as an input character
9. void resetMap()
    a. Will reset the user’s position to (0, 0) on the map
    b. Will initialize Best Buy position to an invalid value (-1, -1)
    c. Initializes npc_count to 0 and positions of all NPCs to the invalid (-1, -1)
    d. Initializes all spots in the 2D map array to empty (‘-’)
10. int getNPCCount()
    a. Returns the number of NPCs currently on the map
11. void setNPCCount(int count)
    a. Sets the number of NPCs on the map to the provided count value
12. int getHackerCount()
    a. Returns the number of Hackers currently on the map
13. void setHackerCount(int count)
    a. Sets the number of Hackers on the map to the provided count value
14. bool isBestBuyOnMap()
    a. Returns true if there is a Best Buy already on the map
15. bool spawnNPC(int row, int col)
    a. Places an NPC on the map at the specified location
    b. Returns false if number of NPCs exceeds npc_count (3)
    c. Returns false if location is invalid
    d. Returns false if location is already filled
16. bool spawnBestBuy(int row, int col)
    a. Places a Best Buy on the map at the specified location
    b. Returns false if location is invalid
    c. Returns false if location is already filled
    d. Returns false if there is a Best Buy already on the map
17. bool spawnHacker(int row, int col)
    a. Places a Hacker on the map at the specified location
    b. Returns false if number of Hackers exceeds npc_count (3)
    c. Returns false if location is invalid
    d. Returns false if location is already filled

For every server room: you will need to initialize the Best Buy location to a random spot in the map. You will also need to spawn between 1-3 random NPCs and 1-3 random Hackers at random spots on the map.

You can use these functions however you like while implementing our map functionality into your game. You can also modify it to better fit your game.

Activities in Each Room:

The goal of the game is to successfully navigate to five server rooms, defeating between 1 - 3 of Carmen's hackers in each room. You must navigate between server rooms using the map.

When displaying the menu, your program needs to provide an option to display a **Status Update**. Here are the categories you need to display:

- The computer's current maintenance level
- How many viruses the computer has
- Computer parts available (list the count of each item individually)
- Antivirus USB sticks available
- VPNs available
- Internet provider level
- How much Dogecoin the player has available
- Player's frustration level
- Carmen's progress
- How many hackers the player has defeated so far

Update the player's frustration level after each action the player takes on the menu: the frustration level should rise by 2 after every action.

In each room, the player may choose between the following actions:

1. Fight a Hacker

a. If you are located in a space with a hacker, you will fight the hacker.
b. See the section on "Fighting a Hacker" to learn about battle gameplay mechanics, including rewards for winning and punishments for losing.
c. If you lose to the hacker, Carmen's progress level increases by 25.
d. After any hacker battle, no matter the outcome, the computer's maintenance level has a 30% chance to drop by 10.
    i. HOWEVER, if this is the final hacker battle in room 5, and they win, do not drop their maintenance level.

e. After any hacker battle, no matter the outcome, the player will lose a random computer part from their inventory.
    i. HOWEVER, if this is the final hacker battle in room 5, and they win, do not take a random computer part from their inventory.

2. Speak to NPC (non-player character):

a. There are mysterious NPC in every room. If you complete their puzzle, they will give you a free, randomly chosen computer part. See section on "Completing Puzzles" to learn how the puzzle gameplay mechanics' work. However, if you don't want to complete their puzzle, you can "Take your chances" and see if they are your friend. There is a 33% chance that the NPC will be a friend and willing to give you a randomly selected computer part for free.
b. There is a 33% chance that the NPC will be neutral. Nothing special happens here.
c. There is a 33% chance that the NPC will be an enemy. If this is the case, they steal a randomly selected computer part from your inventory.

3. Repair your computer:

a. If you have computer parts, you can choose to repair your computer.
b. When repairing, prompt the player to select which computer part they want to use, as well as how many of that part they want to use. See the "Computer Repairs" section to learn the chance of a successful repair.
c. There is a 5:1 ratio between the number of computer parts used and the increase in the computer's maintenance level. If the player's computer is entirely broken, it will require 5 parts to be fixed.

4. Use your antivirus software: The player can choose to use a USB stick of antivirus software to rid their computer of viruses. See the section on Viruses for more information.

5. Travel the Server Room

When the player moves around the server room, there is a 30% chance of a misfortune after each step.

a. If a misfortune occurs, the player loses a random computer part. If they don't have any computer parts, their computer maintenance level is dropped by 10.
b. Once the player defeats between 1 - 3 hackers in a server room, **an option to move to the next server room will be added to the menu.**
c. See the "Map Display" section to learn about how to navigate around the map.

6. Browse StackOverflow: The player can choose to browse StackOverflow in an effort to lower their frustration levels. When the player chooses this option, they must successfully solve a puzzle or win a game (Such as rock, paper, scissors) in order to reduce their frustration. The player’s frustration level drops by 5 points each time they solve a puzzle or win a game. See the section on Puzzles for more information.

7. Quit: If the player chooses to quit, the game ends. Print a message expressing regret that the player could not win.

Misfortunes:

There is more to the game than fighting hackers and exploring server rooms. At every turn in the game, after the player chooses and completes one activity (exploring, fighting, solving puzzles, talking to npcs), there is a **30% probability** a misfortunate event might occur.

If a misfortune occurs at the end of a particular turn, choose one misfortune, at random, between the following options:

a. The player is robbed of computer parts or antivirus software, chosen randomly. Print a message announcing the misfortune:
```
OH NO! Your team was robbed by Carmen’s dastardly
hackers!
You have no computer parts/antivirus software
left!
```

b. The player’s computer is damaged. Reduce the computer’s maintenance level to 10 and print a message announcing the misfortune:
```
OH NO! Your computer was damaged!
```

c. The player's frustration level is increased by 10. Print a message announcing their increased frustration.
```
OH NO! Why won’t my code work!!!!
Your frustration level was increased.
```

Note: If the player’s frustration level rises to 100 at the end of a turn, they will rage quit. If the player rage quits, they lose the game. Announce their rage quitting like so:
```
OH NO! You have rage quit!
Looks like you couldn’t handle Carmen’s hackers.
```

Note: If you want to implement “positive events” you are free to include those as well. For example: “The player discovered some spare computer parts. You gained 2 GPUs and 1 power supply unit.”

Computer Repairs:

Repairing a computer

The player may choose to repair their computer if its maintenance level is less than 100. When a player repairs their computer, your program should first list the player's inventory, and then prompt the user to select which part they would like to use in the repair, and how many of that part they would like to use. Allow the player to select up to 5 parts during a repair. The player may attempt to repair their computer with fewer parts if they so desire. See below for an example of what your program's output might look like during this step.

```
INVENTORY:
CPU: 1
GPU: 0
Power Supply Unit: 3
Computer Case: 2
Internet Card: 0
Keyboard and mouse: 1
Enter the name of the computer part you would like to use:
```

There is a 5:1 ratio of computer parts to a fully repaired computer. Use the following guidelines when calculating how much the computer's maintenance level should improve after a successful repair:

Parts Used	Maintenance Level Increase
1	20
2	40
3	60
4	80
5	100

Note: that the player may not repair their computer if it has a virus. See the section on Viruses for more information.

Fighting a Hacker:

No two excursions through the server room are exactly the same. Players come into contact with different hackers based on which server room they are in. Every room has hackers that only dwell in that room. This is determined by the challenge level of the hacker's computer. Challenge levels 1-5 correspond to the five rooms where challenge level 1 is the Turing Room, challenge level 2 is the Ellis Room, etc., all the way to challenge level 5, which is the Moore Room. For example, the Turing Room only has the following hackers who have a challenge level of 1: hansolo1337, gandalf5000, tomato_cultivator20, cheeto_dweller.

Information about each hacker can be found in the file "*hackers.txt*". See the Hackers Text File section for specific details.

When a player chooses to fight a hacker, there will be an equal probability of which hacker the player will fight. For example, in the Turing Room, you have an equal chance of either fighting hansolo1337, gandalf5000, tomato_cultivator20, cheeto_dweller. If a hacker is successfully defeated, then you cannot encounter that hacker again.

When a player encounters a hacker, print a message announcing the event and print out the name of the hacker encountered:

```
You just ran into lizzathealien! Think you can beat this
                    hacker's skills?
```

The player will then be given two choices:

1. **Attack:**

- The player can attempt to fight the hacker. The success of the battle depends on an equation with different variables, the values of which are based on the player's stats.
- Calculate the following variables:
    - o i = internet provider level that the user owns
    - o v = # of VPNs the player owns
    - o c = the room number of the hacker
    - o r_1 = random number from 1 to 6
    - o r_2 = random number from 1 to 6 (independent of r1)
- Using these variables, use this equation to determine the outcome of the battle:

$$(r_1 i) - (r_2 c \cdot \frac{1}{v})$$

- If the result of the equation is greater than 0, the player wins the battle. If the result is less than or equal to 0, the player loses.
- If they win the battle, the player gains 50 Dogecoins.

- If they lose the battle, the player loses 20 of their computer maintenance level. The player's computer also has a 10% chance of being infected with a virus. See the section on Viruses for more information about this mechanic.
- After every battle, the player loses a computer part in the aftermath.

Note: The player can only choose to attack if their computer's maintenance level is above 0,

2. **Forfeit:**

- The player was unable to outsmart Carmen's hacker. The player loses all spare computer parts in their inventory and must continue on. Hopefully they git gud before they encounter another hacker.

Viruses:

When the player loses a hacker battle, they have a 10% chance of getting their computer infected with a virus! When the computer is infected, its maintenance level drops by 10 after each action the player takes on the menu. The player must use one USB stick of antivirus software to get rid of the virus. Each USB stick can only be used once.

Note that it is possible for the player's computer to be infected with multiple viruses. If this happens, the player's computer maintenance level drops by 10 per virus after each action the player takes on the menu. For example, if the player's computer has two viruses, their computer maintenance level should drop by 20 after each action taken.

If the player uses a USB of antivirus software, their computer is wiped clean of all viruses, e.g. if the player has 3 viruses on their computer, they can get rid of all three viruses at once by using a single USB stick.

If the player's computer maintenance level drops below 0 due to a virus, the player loses the game.

Completing Puzzles:

The player must complete a puzzle whenever they choose to engage with an NPC or browse Stack Overflow.

Puzzles are located in an external **puzzles.txt** file, and each puzzle is separated by a line containing ---.

Answers to the puzzles are provided in the **answers.txt** file. The answers follow the same order as the questions, e.g. the first answer in answers.txt corresponds to the first question in puzzles.txt and so on. Each answer is separated by ---.
Each exercise will be either a debugging question or a multiple choice question that tests the player's knowledge of C++ and programming in general.
For the debugging questions, there will only be one error, so the player must enter the line number of the errant line of code as a solution to each debugging puzzle.
For the multiple choice questions, the player must enter the letter that corresponds to the correct answer.

While visiting stack overflow, implement a mini game such as rock, paper, scissors. See https://en.wikipedia.org/wiki/Rock_paper_scissors for the mechanics of this game.

Note: feel free to come up with other puzzles for the player to solve or games for them to beat. Consider the puzzles above an example. You can ask the player to solve anagrams, a system of equations... it's your choice. Make it fun!

Game End:

The player wins the game when they defeat, between 1 - 3 hackers in each of the 5 server rooms.

The player loses the game when:

- The player quits.
- The hackers succeed in stealing all of the information out of the servers (Carmen's progress level reaches 100).
- The player's computer maintenance level reaches 0.
- The player's frustration level becomes 100, and they rage quit!

If the game ends, your program should print a message informing the player of what caused the loss, along with the following: player name, hackers defeated, Dogecoin. You should check each turn to see if the player has lost the game before each action the user takes.

Note: the final stats of the game should also be saved in a file named ***results.txt***

Hackers Text File:

The hackers are described in the file "**hackers.txt**". Each line lists the hacker's name, followed by their server room number, separated by a comma. Here are the contents of the file:

```
hansolo1337,1
gandalf5000,1
tomato_cultivator20,1
cheeto_dweller,1
thejoker33,2
furiosa_11,2
woodythesheriff,2
lizzathealien,2
ironman123,3
pizzaIsKnowledge,3
bing_bong_2015,3
num1_carmenfan77,3
free_almonds,4
uzer_1000,4
twinkle_star99,4
team-3dward-5evr,4
crazyCatLady76,5
numba_1programmer,5
myNameisJeff,5
treatYoSelf,5
```

Minimum Requirements:

Your implementation of Where in the Code is Carmen Sandiego should have:

- 4+ user defined classes (excluding map class)
- A 2D map
- At least two classes with 4+ data members.
    - At least one class must include an array/vector of objects from a class that you created.
- Appropriate methods for each class (including getters and setters)
- Your project implementation must include at least:
    - 6+ if-else statements
    - 6+ loops (while loops, for loops, do-while, in total)
    - 2+ nested loops
    - File IO (both reading from a file and writing to a file)
- Your project must have an interactive component (ask the user for input, create a menu for choices, ...).
- Game stats should be displayed as instructed at each turn. It's more exciting and meaningful! Also, these stats help debug the code.

- The project must make use of the provided Map class. You may choose to write your own 2D Map class or add/remove functionality to/from what's provided to suit your needs.

Work Alone or in a Group no more than two:

For the final project, you are allowed to work with (at most) one other student that is currently in CSCI1300 this semester. You may also choose to work by yourself if you so choose. If you decide to work as a team, you must meet an additional requirement:

Implement a sorting algorithm. Write your implementation of a sorting algorithm (do not use a Library function or any outside resources) and apply it to a task in your program. One situation where the sorting functionality would be useful is for a ranking task, for example: ranking the number of turns it took each player to reach the Final Room.

Note 1: If you work in a team and you do not implement a sorting task, 10 points will be deducted from your point total.
Note 2: We expect that you will be contributing to the project equally. Both group members must submit a zip file for the project, and the solution files can be the same. Indicate your partner's name in comments at the top of each code file. Both partners will book an Interview Grading appointment together and TAs will be grading you individually.

Timeline:

November 5th – 11th: Project design meetings (optional). Meet with professors/TAs to discuss your project ideas and the classes you will create and use. You should bring your class files and code (a list of classes and methods you will implement) to the meeting. During the meeting, we will go over your classes to see if they're feasible and meet requirements.

The project design meeting is recommended if any one of the team members scored below 50/80 on Project2 CodeRunner.

Prior to Friday, November 5th, you must have reserved a slot on **the Project 3 Design Meeting Scheduler** and have the meeting date reserved for a day on or before Thursday November 11th.

Thursday, November 11th, 11:59 pm: Submit class files & Code Skeleton. 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.

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)
}
```

Thursday, December 2nd, 11:59 pm: Final Deliverables. Your project will be due on this date. You must submit a zip file to **Project 3** assignment on Canvas, including all .h and .cpp files. The submission should compile and run. TAs will also be grading comments and style.

Sunday, December 5th, 11:59 pm: Project Report. Write a 1-2 page report containing the following reflection questions:

1. How did you prepare for the project?
2. How did you develop our 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.

The report should be a 1-inch margin, single space, 12pt font size, times new roman. Submit a report as PDF to **Project 3 Report** on Canvas. Every team member should submit an individual report, clearly distinct in content from their teammate.

Project 3 Interview Grading Scheduler:

Interview grading. The project 3 interview grading scheduler will be available before the deadline of this project. The interviews will take place between **December 2nd** and **December 8th**. If you don't sign-up to meet between **December 2nd** and **December 8th** or you miss your interview, then no points will be awarded for the project.

During the interview grading, TAs will be playing your game and asking questions about your game. They will also ask about your implementations and conceptual questions.

Extra Credit (10 points):

Present your project during lecture or through a video.
Lecture: A sign-up sheet will be shared a week before class presentations
Video: Make a 5-minute (+ or - 1 min) video explaining:

- The project idea
- Implementation and approach
- A demonstration of the working project

Collaboration:

All work for this assignment (and of course in general) must be original work. Every project (group or single) must be original and not similar to ANY other source (internet, paid tutor, fellow student, past years projects, etc). You may collaborate and brainstorm with other people, but each assignment submission is to be the unique creation of the author(s). Your code must be original and unique. 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 a group project is found to be non-unique and deemed worthy of being sent to Honor Code Council then BOTH parties are guilty (even if one party decides to "take the blame"). Choose your groups wisely! If you aren't sure if something is okay, then please ask!

Project 3 Points:

Project 3 is 15% of your final grade. Here is a summary of the points.

10 points - for submitting your code skeleton and class files on time.
10 points - submitting the project report as described earlier.

30 points - Implementation requirements.
The minimum requirements are specified on page 13.

50 points - interview grading.
- TA's questions about your project
- Algorithm descriptions, comments, good style
- Code compiles

50 points - project functionality
- The game plays as outlined in the project description
- Your solution accounts for user error

Project 3 total: 150 points

Note: if your code does not compile, you cannot score above 50 points for the project.

Updates
October 4, 2021:

1. Made the misfortune percentages the same and updated the misfortunes
2. Removed the % from Carmen's progress level, computer maintenance level and frustration level.
3. Updated hacker win condition edge cases.

October 6, 2021:

1. Updated the document to allow 3 of each computer part in the inventory to match the computer repair section.