COP 3014 Fall 2021
Homework 7

Total Points: 150 (and 20 points extra credit)
Due: Wednesday 12/01/2021 11:59:00 PM NO EXTENSIONS

1 Objective

The purpose of this assignment is to make sure:

- You are familiar with dynamic memory allocation and two dimensional arrays.
- You are familiar with the concepts of bounds checking and maintaining state.
- You can work with pointers, structures, and files in C++.
- You can approach a complex problem, break it down into various parts, and put together a solution.

For this assignment, please make sure you conform to program specifications and Output requirements. You should now be familiar enough with cout statements and output formatting that you should be able to EXACTLY match the sample output (other than certain exceptions you will be informed about beforehand). You should also have had enough experience with functions to match the function specifications (name, argument list, return type, and functionality) EXACTLY. Failure to do either or both will result in a loss of points.

This assignment requires you to submit several files on Canvas. Please do so in a SINGLE submission. Canvas allows you to turn in multiple files in one submission. Once you have uploaded your first file, click "attach another file" to upload your second file and so on.

Turn in your files `spaceInvaders.cpp` and `pokemonBattle.cpp` to Canvas.

2 Problem 1 - Space Invaders - 70 points

You find yourself on the last line of defense against some aliens bent on taking over the world. To compensate for the lack of human resources to work the cannons, you take a leaf out of the Adam Sandler Playbook and challenge the aliens to a real-life version of space invaders.

For this game, the space above the cannons is partitioned like a grid. You and the aliens agree on a "speed" of advancing - the number of shots the cannons can fire before the aliens advance by 1 spot. The ships also have different levels of shielding. If a shot from any of the cannons lands on an alien ship, they will lose a shield point. If a ship's shield point drops to 0, it will magically disappear. The "game ends when either all ships have disappeared or the ships have reached and landed on Earth.

Specifications

1. Call this program `spaceInvaders.cpp`
2. Write a function called `createGrid`. This function takes the number of rows and columns on the grid as parameters, and returns a pointer to the 2D array of characters that defines the grid.

(a) In the function, create the 2D dynamic array of characters with the given number of rows and columns. (5 points)

(b) First, fill out the grid with '.' to represent empty space. (3 points)

(c) Read in the number of cannons and their column positions from the user. The cannons will always be placed on the "last" row of the grid. On the grid, the cannons are marked by a 'C'. (5 points)

(d) Read in the number of ships, their positions and their shield points from the user. The positions will be marked by their row and column number, assuming position (0,0) is at the top-left of the grid. On the grid, the ships are marked by their shield points. (5 points)

(e) Return the created grid. (2 points)

(f) You may assume that the ship's shield points will only ever be 0-9. So, they can be read in as a char, instead of an int.

(g) You may assume that the user will only enter valid values for the positions. The user entered positions will not force you out of bounds.

3. Write a function called `printGrid` that takes the grid and the number of rows as parameters and returns nothing. This function will print the grid. (5 points)

4. Write a function called `playGame`. This function will take the grid, the number of rows and columns and the number of shots per cannon as parameters and return nothing.

   (a) Count the number of cannons by looking for 'C' in the last row. (2 points)

   (b) Count the number of ships by looking at the non-dot characters in every other row. (2 points)

   (c) Print the grid at the beginning of the game using the `printGrid` function. (1 point)

   (d) Play that game. The following section describes 1 round of the game.

      i. Calculate the number of shots for per round. (2 points)

      ii. Read in the coordinates for each shot. Please note that the coordinates for the shot might not be within bounds. (3 points)

      iii. If the shot landed on a ship, decrease the shield points of the ship by 1. Please note that chars are integer types can can be decremented just like ints. (3 points)

      iv. If a ship's shield points falls to 0, the ship is destroyed and disappears from the grid. (2 points)

      v. Once the defenders have taken all the shots, the remaining ships advance towards the ground by 1 row. (5 points)

      vi. If a ship lands on a cannon, the cannon is destroyed. (3 points)

      vii. Print the grid at the end of the round. (2 points)

   (e) Play the game until it ends one way or another (5 points).

   (f) The game ends on one of 3 conditions:

      i. The Defenders win if all ships are destroyed.

      ii. The Space Invaders win if at least 1 ship reaches the ground.

      iii. The Space Invaders win if all the cannons are destroyed (which means they reached the ground).

5. In main, get the number of rows, columns, and shots per level from the user.

6. Call the `createGrid` function to set up the grid. (3 points)

7. Call the `playGame` function to play the game. (2 points)

8. Make sure to delete the dynamic array after the game ends. (5 points)

9. Make sure your program is commented properly (5 points)

10. You are only allowed the `iostream` library for this program.

Sample Run 1

```
Enter the number of rows in the grid: 4
Enter the number of columns in the grid: 5
Enter number of shots per cannon for this level: 2
Enter the number of cannons: 1
Enter the locations of the cannons: 2
Enter the number of ships: 2
Enter the coordinates of the ships, and its shield points (x,y),SP:
(0,0),2
(1,2),2
Game at the start:
2....
..2..
.....
..C..
Enter the coordinates for 2 shots: (1,2)
(0,3)

Game after round 1
.....
2....
..1..
..C..
Enter the coordinates for 2 shots: (2,2)
(1,0)

Game after round 2
.....
.....
1....
..C..
Enter the coordinates for 2 shots: (2,0)
(1,1)

Game after round 3
.....
.....
.....
..C..
Defenders win
```

Sample Run 2

```
Enter the number of rows in the grid: 5
Enter the number of columns in the grid: 4
Enter number of shots per cannon for this level: 1
Enter the number of cannons: 1
Enter the locations of the cannons: 2
Enter the number of ships: 2
Enter the coordinates of the ships, and its shield points (x,y),SP:
(1,2),3
(0,3),2
```

```
Game at the start:
...2
..3.
....
....
..C.
Enter the coordinates for 1 shots: (1,2)

Game after round 1
....
...2
..2.
....
..C.
Enter the coordinates for 1 shots: (1,3)

Game after round 2
....
....
...1
..2.
..C.
Enter the coordinates for 1 shots: (3,2)

Game after round 3
....
....
....
...1
..1.
Space Invaders Win
```

3 Problem 2 - Pokemon Battle - 80 points + 20 points extra credit

You are a Pokemon Master working for a renowned Pokemon Gym. However, you've had a very busy week, and you don't have a training schedule prepared for this weekend's session. You decide to wing it with a Free for All combat, where the Pokemon in the gym decide to battle each other to see who comes out on top. To make things easier, you decide to write a program to basically let the session run itself. Please write a program that conforms to the following requirements:

1. Call this file `pokemonBattle.cpp`
2. Create a structure called `Move` that contains the following data members (5 points)
    - name = cstring of 50 characters
    - damage - integer
3. Create a structure called `Pokemon` that contains the following data members (10 points):
    - name - cstring of 50 characters
    - type - cstring of 50 characters
    - hitPoints - integer
    - defenceRating - integer

- isAwake - boolean
- moveName - cstring of 50 characters

4. Open the file `pokemon.txt`. The file will be in the following format (5 points):

```
numMoves
move1 damage1
move2 damage2
.
.
.
moveM damageM
numPokemon
name1 type1 HP1 DR1 move1
name2 type2 HP2 DR2 move2
.
.
.
nameN typeN HPN DRN moveN
```

5. The first line of the file is an integer M, representing the number of Moves. The next 'M' lines of the file will consist of 2 pieces of data - the name of the move, followed by the damage the move can do. These are separated by a tab. The next line is an integer N, representing the number of Pokemon. The next N lines of the file will consist of 5 pieces of data - the name of the Pokemon, the Pokemon class, the hitpoints (health) for that Pokemon, the defense rating of that Pokemon and finally the signature move of that Pokemon. This data is tab separated as well.

6. Open the file `session.txt`. This file will be in the following format (5 points):

```
numAttacks
pokemon1 move1 target1 damage1
pokemon2 move2 target2 damage2
.
.
.
pokemonK moveK targetK damageK
```

7. The first line of the file is an integer K, representing the number of attacks. The next K lines, will each consist of the name of a Pokemon, the name of a move, the name of a second Pokemon who is being attacked, and the damage)i points) done by the attack.

8. Read in the number of Moves M from `pokemon.txt`. Create a dynamic array of M Moves, and populate the structure variables with the data from the file. (10 points)

9. Read the number of Pokemon N. Create a dynamic array of N Pokemon structure variables, and populate the structure variables with the data from the file. In the beginning, all the Pokemon are conscious and ready to battle (10 points).

10. Read in the attacks K from the file `session.txt`. (2 points)

11. For each attack, check if the move used is the signature move of the Pokemon using the move. Also check if the damage for the attack is at least equal to the defense rating of the Pokemon being attacked. If both conditions are met, subtract the move's damage from the hitpoints of the Pokemon being attacked. (25 points)

12. If ever the hitpoints of the Pokemon drops BELOW 0, that Pokemon has fainted (5 points).
13. Open the output file `healthy.txt`. Print the names of the Pokemon that are still conscious (not fainted) and the hitpoints they have left, one per line, in the order they appear in the input file, (10 points).
14. Close the files and delete the dynamic arrays (8 points).
15. Make sure your program is commented properly (5 points)
16. You may assume that a fainted Pokemon will not attack any more.
17. You are restricted to the iostream, fstream and `cstring` libraries for this program.

Extra Credit: 20 points

1. Print the output - the Pokemon that haven't fainted with their hitpoints in descending order of remaining hitpoints, instead of the order they appear in the input file. Use the `insertion sort` algorithm as shown below. (15 points)
2. For the sorting, please sort the array of structs and then print the output. Also, please use the given algorithm and not the Bubble Sort algorithm shown in class.
3. Print the message "It was super effective", after the attacking and the attacked Pokemon, to standard output (cout) every time a Pokemon faints. (5 points)

Sample Files

The output shown here includes the extra credit

pokemon.txt

```
9
Take Down 6
Bubble Beam 7
Hyperbeam 10
Thunderbolt 5
Flamethrower 4
Counter 3
Confusion 3
Razor Leaf 4
Poison Sting 5
6
Charizard Fire 17 10 Flamethrower
Gyarados Water 25 16 Hyperbeam
Raichu Electric 27 14 Thunderbolt
Wobbuffet Psychic 14 11 Counter
Bayleef Grass 18 12 Razor Leaf
Bewear Fighting 15 10 Take Down
```

session.txt

```
15
Charizard Flamethrower Bayleef 16
Wobbuffet Counter Gyarados 4
Raichu Thunderbolt Bewear 8
```

```
Bayleef Razor Leaf Wobbuffet 12
Gyarados Hyperbeam Charizard 13
Bewear Take Down Raichu 10
Wobbuffet Mirror Coat Bayleef 17
Charizard Flamethrower Raichu 14
Bayleef Razor Leaf Gyarados 18
Gyarados Hyperbeam Bewear 16
Bewear Take Down Wobbuffet 12
Raichu Thunderbolt Charizard 11
Gyarados Hyperbeam Wobbuffet 15
Bewear Take Down Charizard 18
Raichu Thunderbolt Bewear 13
```

healthy.txt

```
Raichu 23
Gyarados 21
Bayleef 14
Bewear 0
```

Output Printed to Standard Output (only for Extra Credit)

```
Gyarados attacked Wobbuffet. It is super effective
Bewear attacked Charizard. It is super effective
```

Algorithm for Insertion Sort

Please note that the algorithm sorts in ascending order, and is assumed to be working on primitives. You would have to modify it for sorting in descending order and to work with an array of structs. Hint: We cannot compare structs, only individual data members of structs.

```
loop i from 1 to Length(A)
        x = A[i]
        j = i - 1
        loop as long as j >= 0 and A[j] > x
                A[j+1] = A[j]
                j = j - 1
        end loop
        A[j+1] = x
end loop
```

4 Generic Guidelines

1. Include the header comment with your name and other information on the top of your files.
2. **This is individual work. You may NOT collaborate with other students in the course, former students, hire tutors to "help", copy solutions off the internet, or use pay-for solution websites, including but not limited to Chegg, CourseHero, WiseAnt, Bartelby, tutor.com, assorted Social Media groups, whatever GroupMe students might have created, etc.) This includes posting the problem statements on these websites even if you do not use the answer obtained. Doing so is a violation of the Academic Honor Code. Violation of the Honor Code will result in a 0 grade on the program, a reduced letter grade in the course and potentially more serious consequences.**

3. Please make sure you're only using the concepts already discussed in class. That is, please try and restrict yourself to input/output statements, variables, selection statements, loops, functions, arrays, strings, pointers and structures. Using any other concept (like classes, C++ STL data structures, or C++ 11 style iterators) will result in loss of points.
4. The first problem is worth 70 points and the second is worth 80 points.
5. If we have listed a specification and allocated point for it, you will lose points if that particular item is missing from your code, even if it is trivial.
6. No global variables (variables outside of main()). constants and symbolic constants are allowed.
7. No `goto` statements or the `auto` keyword.
8. No `break` or `continue` statements. The only exception is the use of the `break` statement to prevent the fall-though in a `switch` statement.
9. Functions have to be declared above main and defined below main. Not doing so will result in a loss of 10 points.
10. This assignment requires the use of dynamic memory allocation. If a program crashes with a segmentation fault, or memory leak, it will result in a loss of 10 points per issue. Please test extensively.
11. Issues with dynamic memory allocation could result in PERMANENT issues on your computer. Please make sure you develop and test using CLion, which is protected against those issues.
12. All input and output must be done with streams, using the library `iostream` or `fstream`
13. You may only use the `iostream, cstring` and `fstream` libraries for programs (you do not need any others for these tasks). You are also not allowed to use C++ string objects. Using other libraries will result in a loss of 10 points.
14. Please make sure that you're conforming to specifications (program name, function names, print statements, expected inputs and outputs etc.). Not doing so will result in a loss of points. Please note that names are case sensitive.
15. NO C style printing is permitted. (Aka, don't use printf). Use cout if you need to print to the screen.
16. When you write source code, it should be readable and well-documented (comments).
17. Make sure you either develop with or test with CLion (to be sure it reports no compile errors or warnings) before you submit the program.
18. Testing your program thoroughly is a part of writing good code. We give you sample runs to make sure you match our output requirements and to get a general idea of how we would test your code. Matching your outputs for JUST the sample runs is not a guarantee of a 100. We have several extensive test cases.
19. Please make sure you've compiled and run your program before you turn it in. Compilation errors can be quite costly. We take 5 points off per compiler error for the first 9 errors. The 10th compiler error will result in a grade of 0.
20. Only a file turned in through Canvas counts as a submission. A file on your computer, even if it hasn't been edited after the deadline, does not count.
21. The student is responsible for making sure they have turned in the right file(s). We will not accept any excuses about inadvertently modifying or deleting files, or turning in the wrong files.
22. **Program submissions** should be done through the Canvas class page, under the assignments tab (if it's not there yet I'll create it soon.) Do not send program submissions through e-mail – e-mail attachments will not be accepted as valid submissions.

23. The ONLY files you will submit via Canvas are `spaceInvaders.cpp` and `pokemonBattle.cpp`.

24. **General Advice** - always keep an untouched copy of your finished homework files in your email. These files will have a time-stamp which will show when they were last worked on and will serve as a backup in case you ever have legitimate problems with submitting files through Canvas. Do this for ALL programs.