Lab04
Deadline: 11:59PM Feb 3

Requirements
In this program, we will write a particle simulator. In this lab, we will write a program that will show as the location of particles after a designated number of seconds.

In this lab, we are assuming that we are working on an XY grid that is 20 by 20. Your particles will only move within this boundary. The grid is boxed by borders. The vertices of the borders are at (-1,-1), (-1,20), (20,-1), and (20,20). You are given an input file which contains a set of (x, y) coordinates and (x,y) velocities. These coordinates are initial positions and velocities of the particles. The unit of velocity is 1 unit per second. For example, let's assume that input.txt contains the following inputs:

```
>> cat input.txt
0,0,1,0
0,1,1,0
1,1,1,0
```

In this case, we have 3 particles located at (0,0), (0,1), and (1,1). Their x-direction initial velocity is 1 for all particles and -direction initial velocity is 0 for all particles. After one second, the following are their new respective coordinates: (1,0), (1,1), and (2,1).

Particles can only move in x-axis, y-axis or 45 degree angle directions. In case of 45 degree angle directions, the x and y directional velocity will have the same magnitude. A couple of example velocities are (1,1) and (1,-1).

Your program will simulate this movement and output the final position in the output file. For the above example, if we ran the simulation for 3 seconds, the final output positions will be the following: (3,0), (3,1), and (4,1).

Your output file must be a graphical position of particles with borders. Below is the sample output file for 5 by 5 grid with (3,0), (3,1), and (4,1) coordinates:

```
>> cat output.txt
*******
*     *
*     *
*     *
*   ++*
*   + *
*******
```

Note that the output is shown for 5 by 5 as an exemplary purpose only. Your program must be able to print 20 by 20 grid. Your border must be marked with * symbols while your particles must be marked with + symbols.

Your particle can bound off the border and you should account for those. We assume that the particles are soaked into the border for a second and bounces back. For particles moving in x and y directions, the bounding just reverses the direction while keeping the same magnitude of the velocity. For example, (1,0) with -1 x-direction velocity will be at (0,0) after one second, (-1,0) after two seconds, and (0,0) after three seconds. Any particle in the border such as at (-1,0) will not be displayed in the output. We consider that these are hidden.

For those traveling at 45 degree angle, the bounding will reflect the angle. Here is a particle with (1,1) with 1 x-direction velocity and -1 y-direction velocity. The position will be (2,0) after one second, (3,-1) after two seconds, and (4,0) after three seconds. There can be a case where the particle can be at (0,0) moving at -1 x-direction and -1 y-direction. This particle will be at (0,0) initially, (-1,-1) after one second, (0,0) after two seconds and (1,1) after three seconds.

Finally, these particles have a property where if they collide, they disappear. Let's assume that we have two particles with following properties:
(1,3) at 1 x-direction velocity and 0 y-direction velocity
(2,2) at 0 x-direction velocity and 1 y-direction velocity

These two particles will both be at (2,3) after one second. Since they are at the same coordinate at the same time, they are considered collided, so these two particles will be removed from the system. This means these two particles do not exist anymore. Note that particles never collide when they are soaked in the border. Here is an example with initial positions:
(1,0) at 1 x-direction velocity and -1 y-direction velocity
(3,0) at -1 x-direction velocity and -1 y-direction velocity

After a second, both of them will be at (2,-1), but they are in the border, so they do not collide. Therefore, after 2 seconds, their respective coordinates are (3,0) and (1,0).

Now with all information given, your job is to write a program that can simulate this. Your output must be the following format. The example below shows 3 particles, but it is possible that there might be *hidden* particles inside the border. Also, the output below is 5 by 5 and is for exemplary purposes only. Your actual grid must be 20 by 20.

```
>>cat output.txt
*******
*     *
*     *
*     *
*  +  *
*  ++ *
*******
```

Detailed specifications are below.

1. The command line argument is in this order <executable> input_file output_file num_sec
2. Your output contains the location of particles after the specified number of seconds from the command line have elapsed.

3. Your program must not print anything to the command line. Failure to do so will result in the penalty of -3 points.
4. You will write a main function as well as other helper functions.
5. Your input file is in X,Y, X_velocity, Y_velocity format and order. There will be no spaces.
6. You are guaranteed to have particles inside 20 by 20 grid. Your velocities will be within the range of -5 and 5 inclusive.
7. There will not be any corner case testing.

Restrictions

- Only use standard libraries listed in Learning Hub

Grading

Any grading failure due to not following specifications will result in 0. For full marks this week, you must:

- (2 point) Correctly submit A number file
- (2 point) Not having any files in github other than lab4.c and AXXXX.txt
- (3 point) Generate a correct solution to the problem(s) in this lab

Submission Files

- You must deliver only one .c file named: **lab4.c**
- The file that you submit should be a .c file (not .txt, not .cpp or any other type)
- lab4.c (do not capitalize)
- AXXXX.txt (empty file, but with your A number as file name. Make sure to include 0's, match this A number with your A number in learning hub, and have .txt extension)
- In Github, there must be only lab4.c and AXXX.txt file. You must have 2 AXXX.txt files since it is a group of 2. Nothing else, so make sure to remove all other files there.
- Github: https://classroom.github.com/a/qtnfIXpD