CPSC 350: Data Structures and Algorithms
Spring 2025
Programming Assignment 2: Not So Super Mario Bros.
Due: See Canvas for the due date

Please fill out this form if you have any inquiries on your PA grades. This request will go to our grader. If you have a personal question, please message your instructor first, and they'll let you know how to proceed.

Want to collaborate on code with your partner?

LiveShare tutorial for VSCode → LiveShareTutorial.mp4
If you work in pairs:

- Both students need to turn in a copy of the program
- Both students should cite each other in their README.txt files AND the header comments (see bottom of these instructions)

Sample Files

- Sample input file
- Sample output file

The Assignment

Now that we have gone through the trouble of discussing the "theory" of arrays, we might as well put them to good use in what we hope will be an interesting assignment that will allow you to show off your programming skills.

Your chore for this programming assignment is to implement a Super Mario Bros. inspired simulation, which should give you the opportunity to work on a non-trivial program and have some fun at the same time.

Gameplay

Our Not So Super Mario Bros. simulation is a far cry from the original game, but the goal remains the same – defeat Bowser and save the princess.

The game simulation is carried out in a world that consists of a fixed number of levels. Mario must navigate the levels, encountering enemies and power ups, and collecting coins, before engaging a boss to move onto the next level. A general outline of the program follows below.

The World

The world consists of a number of levels, L, which will be provided at runtime. To finish the game, Mario must navigate from level 1 to level L by defeating intermediate bosses and defeat the final boss in level L. If Mario runs out of lives before this task is accomplished, the game is lost.

Mario: Our Protagonist
In our simulation, Mario starts with a number of lives, V, which will be provided at runtime. Mario also has the ability to collect coins. He starts with 0 coins. For every 20 coins collected, he earns an extra life (and the number of coins resets to 0). At any moment, Mario is in one of 3 power levels, PL0, PL1, and PL2. PL0 is the weakest, and PL2 is the strongest. Mario can increase power levels by coming in contact with a mushroom (a "magic" mushroom) and loses a power level if hurt by an enemy. More on that later.

A Level
Each level in our game is represented by a NxN grid (think 2D array), with N provided at runtime. Mario may navigate the grid by moving up, down, left, and right only. The grid wraps horizontally and vertically (a torus). For example, if Mario is in the rightmost column and moves right, he will wrap to the same row in the leftmost column.

Every position in the grid can be populated with one of the following:

- A coin
- A mushroom
- A Goomba or a Koopa Troopa
- The boss of the level (there must be exactly 1 per level)
- Nothing

Additionally, every level from 1 to L-1 contains a SINGLE warp pipe somewhere on the grid. Note that the final level does not contain a warp pipe.

Mario's Interaction with the Environment

Mario will begin at a random location in the first level, with power PL0, and starts interacting with the environment immediately. Once he has interacted with the current location, he moves on to the next location by moving up, down, left, or right. The direction is chosen randomly with uniform probability (25% each). Interactions with each position occur as follows:

- If the position contains nothing, Mario moves on.
- If the position contains a coin, Mario adds the coin to his wallet, and then the position contains nothing. Mario moves on.
- If the position contains a mushroom, Mario's power level increases by 1, or stays at PL2 if he's already in PL2. Once the mushroom is consumed, the position contains nothing, and Mario moves on.
- If the position contains a Goomba or Koopa, Mario engages per the rules of a regular enemy, below.
- If the position contains the level boss, Mario engages per the rules of a boss, below.

- If Mario encounters the warp pipe, he will immediately jump to a random position in the next level without having to beat the current-level boss.

Mario's Interaction with a Regular Enemy

When Mario encounters a regular enemy, gameplay proceeds as follows:

- If the enemy is a Goomba, Mario defeats the enemy with a 80% probability, the position becomes empty, and Mario moves on. If Mario loses (20% probability), his power level is decreased by 1, the Goomba stays in the position, and Mario moves on. The exception is if Mario loses while at PL0, in which case a life is lost. If Mario has another life, he continues at the same location, with PL0. Otherwise, the simulation ends, and the game is lost.
- If the enemy is a Koopa, Mario defeats the enemy with a 65% probability, the position becomes empty, and Mario moves on. If Mario loses (35% probability), his power level is decreased by 1, the Koopa stays in the position, and Mario moves on. The exception is if Mario loses while at PL0, in which case a life is lost. If Mario has another life, he continues at the same location, with PL0. Otherwise, the simulation ends, and the game is lost.
- If Mario defeats 7 enemies on the same life, he earns another life.

Mario's Interaction with a Boss

When Mario encounters a level boss, gameplay proceeds as follows:

- Mario defeats the enemy with a 50% probability, the position becomes empty, and Mario moves on to the next level. If Mario is in the last level, the princess is saved, the game is won, and the simulation is over. If Mario loses (50% probability), his power level is decreased by 2, and Mario attempts to defeat the boss again. The exception is if Mario loses while at PL0 or PL1, in which case a life is lost. If Mario has another life, he continues at the same location, with PL0. Otherwise, the simulation ends, and the game is lost.

The Program

Your program will take as command line input the name of a plain text file that has the following format:
Line # 1 is the number of levels, L (a positive integer)
Line # 2 is the dimension of the grid, N (a positive integer)
Line # 3 is the number of initial lives, V (a positive integer)
Line # 4 is the approximate percentage of the positions in each level with **coins** (a positive integer)
Line # 5 is the approximate percentage of the positions in each level with **nothing** (a positive integer)
Line # 6 is the approximate percentage of the positions in each level with **Goombas** (a positive integer)

Line # 7 is the approximate percentage of the positions in each level with **Koopas** (a positive integer)
Line # 8 is the approximate percentage of the positions in each level with **mushrooms** (a positive integer)

Note that the values on line 4-8 MUST sum to 100.

Once the file is read, a world with the appropriate number of levels is constructed. Each level should be populated using the specifications provided. In addition, a level boss should be placed randomly, as well as a warp pipe (for all but the last level). Mario is placed at a random position in the first level with PL0, and the simulation begins until the game is won or lost.

The output of your game is a plain text file that represents a log of every move Mario has made. The name of the log should be provided as a command line argument.

The log should start by printing a character representation (see below) of each level, prior to Mario being placed and interacting with the environment. There should be a newline after each level representation.

Each subsequent line of the file should specify:

- The level number
- The position location of Mario (a row and column number)
- The current power level of Mario before interacting with the position
- The action that was taken at that position
    - Mario collected a coin
    - Mario ate a mushroom
    - Mario fought a Goomba and won/lost
    - Mario fought a Koopa and won/lost
    - Mario fought the level boss and won/lost
    - The position is empty
    - Mario warped
- The number of lives Mario has after interacting with the position
- The number of coins Mario has after interacting with the position
- The direction that Mario will move next (UP, DOWN, LEFT, RIGHT,STAY PUT)

After each line, a character array representing the current level should be printed, followed by a newline.

The last line of the file should specify whether Mario won or lost the game, and the total number of moves it took to reach that point.

Data Structure

You shall represent your world (textually) as a 3D LxNxN array of characters. For each of the L levels, each of the NxN grid positions are represented by the following characters:

- x - nothing
- m - a mushroom
- c - a coin
- g - a Goomba
- k - a Koopa Troopa
- b - the level boss
- w - a warp pipe
- H - stands for hero. Mario's current position.

Rules of Engagement

- You may **NOT** use any non-primitive data structures (eg. vectors, lists, etc) other than arrays. Of course, to do the file processing you may use any of the standard C++ IO classes.
- For this assignment, you may work in groups of NO MORE THAN TWO. Group members will receive the same grade regardless of individual contributions, so choose wisely.
- Develop using VSCode and make sure your code runs correctly with g++ using the course docker container.
- Feel free to use whatever textbooks or Internet sites you want to refresh your memory with C++ IO operations, just cite them in a README file turned in with your code. All code you write, of course, must be your own. In your README please be sure to include the g++ command for compiling your code.

Due Date

This assignment is due on the day and time specified on Canvas. Submit all your commented code as a zip file to Canvas. The name of the zip file should be LastName_FirstInitial_A2.zip

Grading

Grades will be based on correctness, adherence to the guidelines, and code quality (including the presence of meaningful comments). An elegant, OO solution will receive much more credit than procedural spaghetti code. I assume you are familiar with the standard style guide for C++, which you should follow. (See the course page on Canvas for a C++ style guide and Coding Documentation Requirements.)

Again, code that does not follow the specification EXACTLY will receive an automatic 25% deduction. Code that does not compile will receive an automatic 50% deduction.

Readme & References

More info on Canvas!

- **README file:** All source code will be accompanied with a plain text README file. We encourage students to take advantage of markdown technology if they are inclined to do so. This file will contain:
    - The following identifying information:
        - Full name
        - Student ID
        - Chapman email
        - Course number and section
        - Assignment or exercise number
    - A list of all source files submitted for the assignment
    - A description of any known compile or runtime errors, code limitations, or deviations from the assignment specification (if applicable)
    - A list of all references used to complete the assignment, including peers (if applicable)
    - Instructions for running the assignment. (Typically applicable in advanced courses using build systems or third-party libraries) for us it would be something like:
        - To compile: g++ *.cpp -o A1.exe
        - To run: ./A1.exe input.txt output.html
- **In-code citations:** You are allowed to use small, isolated lines of code from external sources in your programming assignments as long as they are appropriately cited. Any time you are including code that you did not write yourself, it must be cited. This includes sources from StackOverflow and similar forums, other current or previous students, tutors, books, tutorials, etc. You should wrap the copied code in a comment denoting the start and end of said code, like this:

```
# code that is not copied (you wrote this)
someCode = 0;
print("hi I wrote this!")

"""BEGIN CODE FROM SOURCE: link/name of source"""
print("I did not write this");
"""END OF CODE FROM SOURCE: link/name of source"""

"""BEGIN CODE FROM CHAT GPT, PROMPT ASKED: how do you ...?
"""
print ("I did not write this");
"""END OF CODE FROM CHAT GPT"""

# back to code that you personally wrote
codeIWrote = 10;
```

- **Info in code files:** Additionally, all source files will start with a header comment containing the following items (one per line):
    - Full name
    - Worked in partnership with *Full name*

- Student ID
- Chapman email
- Course number and section
- Assignment or exercise number