Due Dec 8 by 11:59pm **Points** 100 **Available** after Nov 12 at 2pm

Home (https://northeastern.instructure.com/courses/156930/pages/gd5d3d707045

Resources (https://northeastern.instructure.com/courses/156930/pages/gd9f1817

Final Project

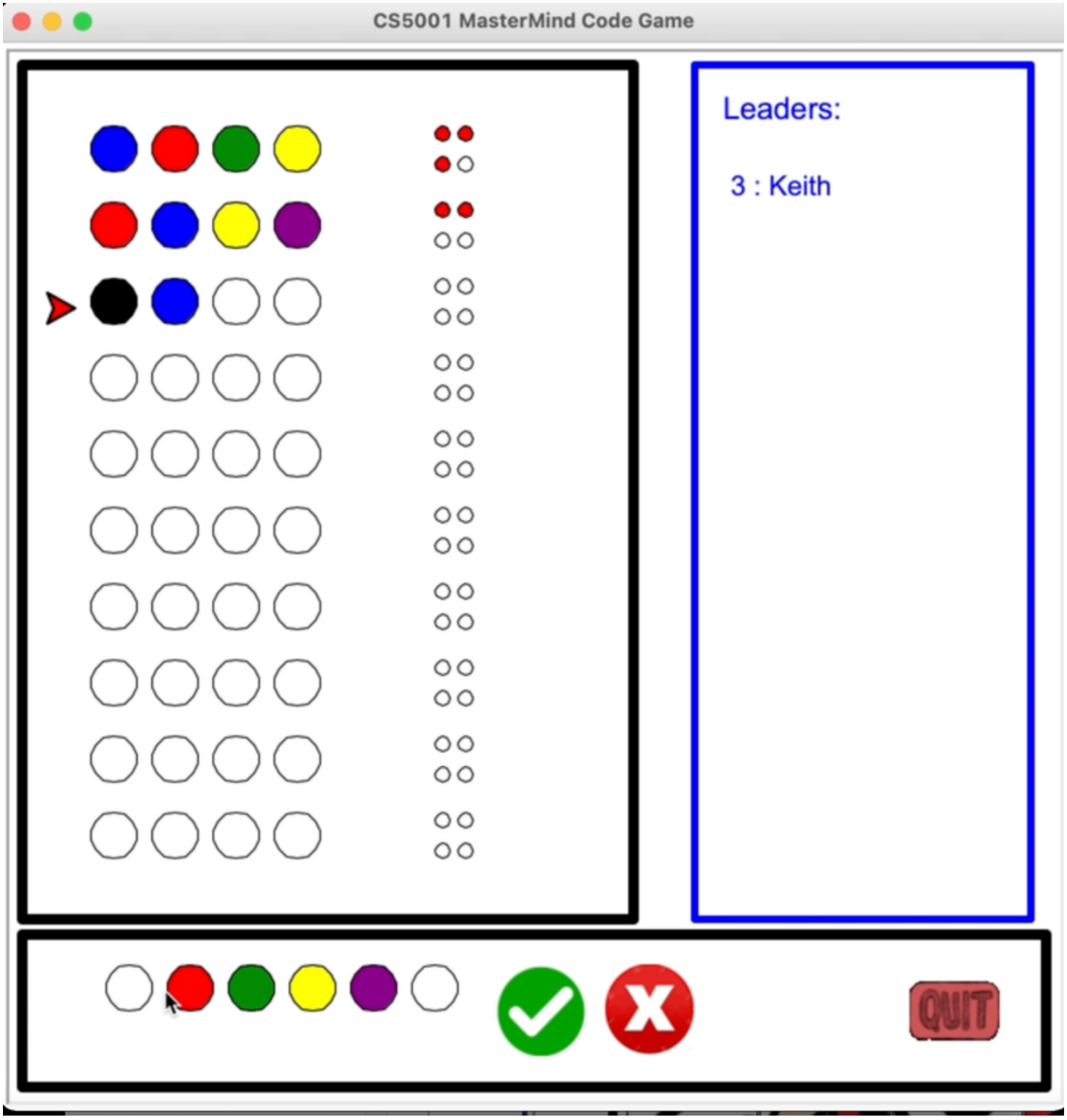
Deadline: December 08, 2023 at 11:59pm ET

Programming Component + Written Design Description (100% of this assignment)

Similar to the homework this semester, submit your project to Gradescope. As always, you may submit multiple times right up until the deadline; we will grade only the most recent submission.

IMPORTANT NOTES:
You may NOT use slip days for this assignment. Any project turned in after the deadline will receive an automatic zero! There are no exceptions to this except University-level approved illness or family crisis (e.g. death, etc.). Manage your time well and be sure to submit something well before the deadline.

We will be using the code similarity matcher on your submissions. **Any form of code sharing with other students, chatGPT (or other AI), or copy-pasting full or partial solutions from the internet will result in an automatic zero for this assignment, an "F" in this course, and an academic integrity report sent to the Registrar.**



The Game: MasterMind

Mastermind (https://en.wikipedia.org/wiki/Mastermind_(board_game)) is a coding-breaking board game for two players. The “secret code” is a set of 4 colors chosen out of a possible 6 colors, where the player needs to guess which colors are in which positions in the 4-color secret code. After each guess, the player learns how many colors in their guess are in the right position, and

how many had a color that's in the code but the color is in the wrong position. Scoring pegs of different colors were used to show them how many correct guesses and how many correct positions they had gotten with their guess. Red pegs meant a correct color but out of position, black pegs meant a correct color in the correct position. These scoring pegs after each guess can be placed in any order – the player never knows _which_ colors are correct and/or in the correct position. Examples of the two-player version of the game being played can be seen

In an older version of Mastermind, called, Bulls and Cows, guesses with correct position were called "bulls", and correct guesses with incorrect positions were called "cows". We will use the "bull" and "cow" terminology when talking about correct guesses for Mastermind.

Our Version of Mastermind: 1 player

We'll be designing a version for one player. In your version, the program selects the 4 color secret code. Your program will also place the scoring pegs. In some versions of Mastermind, blank positions are allowed, and/or duplicate colors are allowed. You can play a one-person version of the game **here (https://www.archimedes-lab.org/mastermind.html)** ; this version allows duplicate colors. Our version will not allow blank positions or duplicate colors. Our permitted colors are:
`colors = ["red", "blue", "green", "yellow", "purple", "black"].`

In the one-player game, the score is the number of guesses it takes the player to guess the code. The player loses if they don't guess the code in 10 tries. Lower scores/fewer guesses are better. Our scoring pegs will be red for cows and black for bulls.

Prof Keith posted walkthroughs and examples of the game play and expected functions for the game, which can be found

- **Here (https://northeastern.instructure.com/media_objects_iframe/m-TQVazQeNYHjYdUKYNyFA5unw1qU35yg?type=video?type=video)**
- **Here, and (https://northeastern.instructure.com/courses/156930/files/24302098/download) (https://northeastern.instructure.com/courses/156930/files/24302098/download?download_frd=1)**
- **Here (https://northeastern.instructure.com/courses/156930/files/24302100/download) (https://northeastern.instructure.com/courses/156930/files/24302100/download?download_frd=1)**

Game Play & Functionality

- We should be able to input the player name to your game through a pop-up window rather

than in the terminal.

- Your program should draw a board as seen in the demo videos. This board for the Mastermind game should include:
  a Leaderboard on the right that lists the previous best scores in the game (lowest scores are best).

- A “Quit” button that exits the game. We provide a gif for this button in the starter code.

- A set of clickable, colored guess buttons that allows a player to choose one color at a time for a guess, and does not allow them to choose duplicate colors. We provide starter code for these buttons in Marble.py

- A green check mark button that confirms that a guess should be checked/entered. We provide a gif for this button in the starter code.
  A red “X” button that removes a guess that has not yet been checked/entered and resets the clickable, colored guess buttons. We provide a gif for this button in the starter code.

- A representation of the board that displays the current guess and previous guesses in this game. We provide starter code for these buttons in Marble.py.

- A part of the display that shows the scoring pegs. These scoring pegs are filled in after a guess is checked, with black pegs indicating colors guessed in the right position, and red pegs indicating colors guessed correctly, but not in the right position. Blank circles can represent neither of those conditions.

- Your program should save the best scores in a leaderboard file, and when the program is re-launched, those scores should be visible with the player name in the leaderboard. A minimum of 2 best scores should be saved. The list should update with new best scores when someone achieves a new best score. Do not submit your leaderboard.txt file with your submission; a new file should be created if one does not exist yet.

- The program should display a visual error message popup if the leaderboard file cannot be found. We provide a gif for this message in the starter code.

- The program should display a visual message popup if the player presses the Quit buttons. We provide a gif for this message in the starter code.

- The program should display a visual message popup if the player wins or loses. We provide gifs for these messages in the starter code.

- All user interaction should be via your turtle-based user interface; users must use the mouse to play your game (the only keyboard actions are when you capture the player name at the start of the game)

- Any errors encountered by the program during use should be written to a text file named "mastermind_errors.err". You have full freedom to determine the schema of the information, but at minimum, we should be able to identify the date/time of the error logged, and the name/type of the error. **You must write this error logging functionality yourself - you may NOT rely on Python's default error logging functions.**

Resources & Guidance

Important: Your project MUST be runnable and do something non-trivial to get credit for this assignment. However, it does not need to be functionally complete to get partial credit.

Turtle is the ONLY Graphic library allowed for this assignment. You may NOT use PyGame, TkInter, or any other graphics package

Resources

This project will have you using a bit of almost everything we learned this semester (except recursion), so be sure to allocate appropriate time to complete it. If you wait until the last minute to work on this, you are almost guaranteed NOT to finish on time. You may **NOT** use late days on this project. No submissions will be accepted after the 8th of December. Plan to start early!

At the same time, don't panic. The project is bigger than anything you've worked on this term, but the concepts are all within your reach. You'll need to do a bit of reading on Turtle to brush up on the elements you'll likely need for the graphics. In particular, review the turtle objects themselves and think about how you can use multiples of them to do things on the gameboard for you.

A link to Python Turtle documentation is here:
https://docs.python.org/3/library/turtle.html#turtle.update
(https://docs.python.org/3/library/turtle.html#turtle.update)

Feel free to search the web for other Turtle resources to help you learn whatever you think you

might need to know for the UI portion of your application.

What to Submit

You may submit multiple times right up until the deadline; we will grade only the most recent submission.

NOTE: You may NOT use late days for this assignment. Any project turned in after the deadline will receive an automatic zero! Manage your time well and be sure to submit something well before the deadline.

Your files to turn in:

- `mastermind_game.py` (includes a main() function that lets us play your game)
- `test_mastermind_game.py` (tests for game functions, does **not** test turtle/view)
- `design.txt` (short plaintext file – 2 or 3 paragraphs – explaining your design)

You may turn in other files with helper functions, but these above files are the minimum required.
You are free to select your approach to this problem! The code may be written as classes and objects or procedurally.

Include any and all files required for playing your game, including gifs (either ones we provided for you or custom ones that you created) and any starter code you used, whether you modified it or not.

You may use whatever IDE you've become familiar with during this term (PyCharm, VSCode, Wing are all fine). Note however, that there are sometimes variations of how code runs inside of each of those environments. It is your responsibility to ensure that your solution can run in IDLE as well without modification. Be sure to run your final version of `mastermind_game.py` in IDLE before "shipping" your final submission to us.

Test Suite

Your test file should test at least this: given test data for a "secret guess" and a "simulated user guess" how many correct answers were there? (For this game, your test should identify both "correct color/correct placement" and "correct color/incorrect placement". Ensure you **clearly document** what you've done for your testing in your design.txt file.
You may opt to develop and include other tests you find useful; the "guess test" is the only test

you will be graded and evaluated on.

Starter Code

The starter code I supply is object-oriented (since OO and algorithms were the big final two topics for the semester). **However, you are NOT obligated to use an OO approach for this assignment!** It is perfectly acceptable to create a procedural solution if you wish. You may also create a "hybrid" solution using my OO starter code for some things, and a procedural approach for others. Your have full freedom with your design decisions.

The starter code described below can be found here **Download here (https://northeastern.instructure.com/courses/156930/files/24302099?wrap=1)** ⭳ **(https://northeastern.instructure.com/courses/156930/files/24302099/download?download_frd=1)** .

Using our starter code is optional. You can opt to create all code from scratch if you wish. You can also modify the starter code if you wish. We will be providing you with these files:

`Point.py`
This file contains a class Point with two attributes, x and y. It represents a geometric point with x and y coordinates. It has two methods:
delta_x, which takes as input another Point, and returns the absolute distance of the different between this point and the other point's x coordinates,
delta_y, which takes an input another Point, and returns the absolute value of the difference between this point and the other point's y coordinates.

`Marble.py`
This file contains a class Marble with the attributes pen, color, position, visible, is_empty, and size. This class can draw an empty Marble and set its color, get its color, erase itself, and determine if it has been clicked. This class requires the use of the class Point.

`Many .gif files for your UI (you can create your own if you wish, as long as they are functionally equivalent to what I've provided):`
checkbutton.gif, a green check mark for the "check" button
file_error.gif, a file error popup message
leaderboard_error.gif, a leaderboard access error popup message
lose.gif, a popup message for when the player loses
quit.gif, a red square for the "quit" button
quitmsg.gif, popup message for when the player quits

winner.gif, a popup message for when the player wins
xbutton.gif, a red “X” for the “cancel” button
Note: none of the popup messages actually pop up, they’re just gifs that you have to write the code to make them open in a new window, etc.

Other Information & Evaluating Your System

Other Information

All user interaction should be via your Turtle-based user interface; users must use the mouse to play your game (the only keyboard actions are when you capture the player name at the start of the game).

Important Note: We know that Turtle provides a "shell" over some Tk functionality, HOWEVER, you are **NOT** allowed to use **Tkinter**. For the graphical portion of this assignment, you may only import turtle. Any and all other graphics packages are disallowed and will result in a heavy deduction in your project grade if you use them.

Program Evaluation

We will be inspecting your code, playing your game and assessing your overall design

- We’ll be looking for the functionality described in this document and shown in the demo videos for program correctness.
- Document your code properly, following all the guidelines you’ve learned through the semester and what’s in the Style Guide for the course. Yes, that includes docStrings for your functions - all of your functions (you know this already). If you're using an OO approach, be sure to include docStrings for your classes as well.
- Make your code readable, following the style guide with good variable names, good function writing, well-formed and clear classes if you’re using classes, good use of whitespace, and all the other code readability guidelines you’ve learned throughout the semester. Readability and documentation get even more important the longer and more complicated your programs get!
- Keep your code as efficient as possible – don’t do anything twice that you only need to do once, etc. (stay D.R.Y.)
- "UI Aesthetics" and user experience is important as well. Turtle is NOT a real-time graphics system so we won't do anything to purposely overload the event system. However, your game should "look and feel" like a proper MasterMind game. If your screen does not update properly, or if you are not rendering graphics correctly, you will have marks deducted for

problems with your UI Aesthetics

Important: **Your project MUST be runnable and do something non-trivial to earn credit!** However, your program does not need to be functionally complete to get partial credit. Having something **non-trivial** running and working gets you more credit than having a bunch of code that doesn't work at all. Code that runs but is trivial (just so you can say "I have something working") will earn you a zero on this project.

Pro-Tips

Breathe!

This is a big project with a lot of moving pieces, but everything I'm asking you to do is well within your skill-level. Remember our concepts on procedural (or object) decomposition and take things one chunk at a time. You've got this. You'll be fine. Here are some tips to help you crush this...

Manage Your Time!

This project will have you using a bit of almost everything we learned this semester (except recursion), so be sure to allocate appropriate time to complete it. If you wait until the last minute to work on this, you are almost guaranteed NOT to finish on time. **You cannot use late days on this project**, because we have a deadline for turning in grades. Plan to start early!

At the same time, don't panic. The project is bigger than anything you've worked on this term, but the concepts are all within your reach. You'll need to do a bit of reading on Turtle to brush up on the elements you'll likely need for the graphics. In particular, review the turtle objects themselves and think about how you can use multiples of them to do things on the gameboard for you.

A link to Python Turtle documentation is here: **https://docs.python.org/3/library/turtle.html#turtle.update** **(https://docs.python.org/3/library/turtle.html#turtle.update)**

Feel free to search the web for other Turtle resources to help you learn whatever you think you might need to know for the UI portion of your application.

I said it before, but must emphasize this point:

> ***** DO NOT WAIT UNTIL THE FINAL WEEK TO START THIS PROJECT! *****
> **I promise you, you will not be able to complete this project if you wait until the last minute.**

You have about 3.5 weeks to complete this assignment. Work a little bit EVERY day (even if it's refactoring your code) to make progress.

Have a plan and work towards smaller milestones to keep your development on pace for completion. If you haven't planned a project before, a sample work plan is below. Feel free to use it as a rough guide if you're not sure how to get started. (You'll need to act as your own project manager and writing down some timeboxed goals will likely be helpful).

Sample Work Plan

Here are some of the things I did when creating the sample solution - you can adjust these to suit your working style and progress.

Milestone 0: *Turtle Pre-work Exploration* (In agile terms, this would be called a "spike" where we explore any technology we're a bit shaky on, so we can learn & gain confidence before trying to use it).

Goal: Review how to draw shapes, capture mouse clicks, use turtle objects to render .gif images in the proper locations, etc. If you're like me, you may have forgotten some of the Turtle functionality since the beginning of the semester. Take a day to practice and write code to do the basic drawing and event handling you'll need. Turtle also has some rudimentary "dialog box" functionality like what you see me use in the video to get the player name and the maximum number of moves allowed. It's pretty straightforward, but you'll need to read the documentation to see how to use it. Some of this code will be "throw-away" but it's a learning exercise before digging in.

[1-2 days]

Milestone 1: Get a Text version working first. Goal: Implement the fundamentals for basic game without worrying about the graphics yet. Develop the functionality to generate a randomized "secret code" and the ability to test user input via count_bulls_and_cows(). Don't spend a lot of time with an elegant textual interface; you just want to get the core of the "game model" working so you can use it later.

[1 day]

Milestone 2: *Develop Gameboard:* Write the code to create the entire game board: The play area, status area & leader board. Validate the code works with "dummy" data.

[1 day]

Milestone 3: *Marble placement & behavior*: Write the code to manage proper Marble placement for the game AND the Marble “Guess Panel” (at the bottom of the screen where the user selects the colors they want). If you’re using our included Marble code, you’ll likely want to spend a little time examining that code and seeing how to use it. Check your layout algorithm to make sure it works for placing each User Guess in the correct spot. Implement code to handle mouse clicks & selecting Marbles appropriately.

[3 days]

Milestone 4: *Game behavior:* Write the code to implement the game rules. Generate the secret code. Accept clicks on Marbles to guess positions. React appropriately to the Reset button (to cancel current color selections) and Quit button to exit the game.

[3 days]

Milestone 5: *Leaderboard:* Write the code to implement saving/retrieving and showing real contents of the leader board. Continue to test.

[1 day]

Milestone 6: *Clean up:* Refactor any code needed to handle issues with rendering. Clean up code, continue to test.

[1 days]

Milestone 7: *Error logger:* Write the code to log errors to mastermind_errors.err. Clean up code, continue to test.

[1 day]

Milestone 8: *Write design.txt:* Write your design description. Do your final testing and any extra optional work (e.g. skinning your game). Wrap up any other outstanding task.

[1 day]

Release: Before 11:59pm, December 08th: Make a "staging area" with all the code and assets you need to run your game. Run the game ONE final time to make sure it works and you haven't left anything out. Ensure the staging area has your design.txt document too (that's part of your grade). Double-triple check you have everything required for launch and - *Ship It!*

After You're Done

Kick your feet up and relax! Share the game with friends and family so they can spend their copious free time playing MasterMind, thereby freeing your Netflix account so you can watch *Pluto* (*Scavengers' Reign* is good too, if you have MAX)!

Oh, and don't forget to celebrate a job well done! 😀