Introduction

In this homework, we are going to do something a little different than the rest of the homework assignments that you have done so far. Many times in Computer Science, you will receive a strict set of criteria that tells you exactly how you should solve the problem– just like we have done all semester. In other cases, (the more fun cases) you will only be given general guidelines to follow, and it is up to you to use your creativity and (newly developed) problem solving skills to get the job done! In homework 13, we want to tap into your creativity and allow you to pick your own project that you can make your own.

In addition to making the project your own, we also want to introduce you to doing your own research while trying to solve computer science problems. Outside of the classroom, you will not always be told which functions to use and how to use them, and that is exactly what this project is all about. We are not going to tell you how to solve the problem – that is up to you. **We may give a few hints and guidelines, but the implementation is completely up to you. There are no banned functions.**

So get out there, be creative, and have fun with this project!

Grading

Each project will be graded out of 100 points. There is a total of 100 extra credit points available for each project (yes, 100 points). The extra credit is more challenging and will push your problem solving skills! **Note** that this project will NOT be autograded. It will be hand-graded by the TAs, so worry less about exact formatting and worry more about impressing the TAs with your awesome projects.

What to turn in

Each project below lists the specific things that you must turn in if you choose that project. **However, each project must also include a text document with answers to the following questions:**

1. What project did you choose? Why did you choose that project? (1-2 sentences)
2. How did you solve the problem? Why did you solve it that way?
3. What did you learn?

Your responses to these questions don't have to be very long. Focus on answering the question succinctly and accurately. **Note that the text document is not graded directly, but your project will not be graded if you do not submit the text document**. The text document should be named <gt_username>_analysis.txt where "<gt_username>" is replaced with your GT username, not your GTID.

Table of Contents

There are a total of 5 projects to choose from. Check out the links below to find the description of each. **You only have to choose one!**

1. Write your own autograder
2. Become an Excel master
3. Learn about 3d animation and build your own working clock
4. Learn about GUIs in MATLAB and make your own 2048 game
5. Build your own website

Project Option #1: Write your own autograder

Motivation:

You are nearing the end of a wonderful semester at Georgia Tech, when a terrible catastrophe happens – Georgia Tech servers catch fire and are irreparably destroyed. All Computer Science classes, including your beloved CS 1371, are in chaos. Since the TAs can't possibly grade all the remaining homework submissions by hand, they tell the class that everyone will get a 0, *unless* someone can come up with a quick fix. You realize that, given all your MATLAB knowledge, you yourself can create a rudimentary autograder!

Description:

Create a function, `autograder.m`, that takes in a rubric structure, and is placed in the same directory as student code, and autogrades their homework. You can assume that you're in a folder that contains the following files and directories:

- A folder with solution files (.p)
- A folder for each student containing the homework files that the student turned in (ie, `rectangleMath.m`)

Your job is to generate feedback for each student. The student folders are named based on the student's name; for example, if the student's name is Omar Ahmed, their folder is "Omar Ahmed". You can assume that each student has a folder and their code won't error. You can assume that all students have a folder.

The rubric is a 1x1 structure that contains information about each problem:

`rubric:`

```
name: <The name of the problem>
testCases: <Cell array of test cases for that problem>
grades: <Point values assigned for that problem>
bannedFunctions: <Cell array of banned functions (for extra credit only!)>
```

`name` is a string that represents the name of the problem; for example, if the problem is called `getFeedback`, then the name field of rubric will have `getFeedback` for that problem.

`testCases` is a cell array where each value is the complete set of inputs for the given test case. For example, if the first test case has three inputs (1, 2, 3), then `testCases` would have `{{1, 2, 3}}` as its value. The length of `testCases` is the number of test cases, and each test case is guaranteed to be a cell array; that is, `testCases{1}` will be the complete set of inputs for the first test case.

`grades` is a vector of numbers, the same size as `testCases`. Each value in `grades` is the number of points to assign for that specific test case; for example, if the first test case was worth 15 points, and the second test case was worth 5 points, `grades` would be `[15 5]`. You are guaranteed that all points will always add up to 100 percent.

`bannedFunctions` is a cell array of strings that represent functions the student should not be able to use. For example, if you shouldn't be allowed to use the `numel()` function, then `bannedFunctions` would contain `{'numel'}`.

`rubric` is a structure array; below is an example of what `rubric` might look like:

```
rubric(1):
      name: drinkWater
      testCases: {{1, 2, 3}, {55, 6, -7}, {45, 44, 1}}
      grades: [10 10 20]
      bannedFunctions: {'imread', 'why'}
rubric(2):
      name: theRest
      testCases: {{[1 2 3]}, {'helloWorld1'}, {}}
      grades: [10 15 35]
      bannedFunctions: {'size', 'numel'}
```

Your autograder should create some kind of feedback file (whether it be text or HTML or anything else you can come up with) that contains the following information:

- Student name
- List of each problem and test case
- Whether they got the test case right or wrong
- The points the problem was worth
- Some kind of summary of their grade for each problem
- Some kind of summary of their overall grade

Each student should have a feedback file.

Additionally, we also need to upload these grades to Canvas, so you should generate an Excel file that summarizes each student and their grade for the homework. The exact structure of this file is not important. Think about what would be most useful given the task!

Requirements:

- **(40 points)** Your code successfully loops through all the student folders
- **(30 points)** You correctly write the feedback or each student
- **(15 points)** You correctly write the gradebook Excel file
- **(15 points)** The whole thing works correctly

Extra Credit:

- **(25 points)** You successfully detect and catch all extra outputs (for example, if the function outputs two outputs, you successfully find this and check both outputs for correctness. You are guaranteed that both functions will output the same number of outputs, even if the outputs are wrong).
- **(10 points)** You account for if the student code errors (i.e. your autograder does not crash if a student function crashes).
- **(20 points)** You correctly implement banned functions.
- **(10 points)** You account for the case where the student did not submit the function.
- **(35 points)** You account for infinite loops in the student's code.

Notes:

- You are guaranteed that each student's code will work for the given test case, and that the same goes for the solution code.
- The function name will always be exactly the same as the entry in the `rubric`.
- There will never be any external files used in the code.
- The folder structure is the same as that of the folder provided for this project.

Hints:

- You should look at the functions `cd()`, `dir()`, `feval()`, and `exists()`.
- For the inputs, see what happens when you do this:
  - `a = {[0; 1], [0; 1], 'k'};`
  - `plot(a{:});`
- For `bannedFunctions`, think about which function MATLAB chooses if there's two .m files with the same name? Is it important whether or not a function is in your folder?
- Is there a MATLAB function that can tell you how many outputs a function has?
- You are encouraged to change the provided students' code to test your autograder!

What to turn in:

- Analysis text file (see first page)
- `autograder.m`
- An example feedback file that your function created
- The gradebook file that your function created

Project Option #2: Become an Excel master

Motivation:

In this project, you will step outside of MATLAB land and learn more about Microsoft Excel and how you can use Excel built-in functions while also learning about how MATLAB can help supplement our newly acquired Excel skills.

Description:

For the first part of this project, you will read and follow all of the steps listed in the "Excel_Project.pdf" file. This file will walk you through the basics of excel, as well as dive into some more advanced techniques that will help introduce you to the important topics.

For the second portion of this project, you will need to seek out data that includes at least both double and char types. For example, the following data contains both double and char types:

Fruit	Rating	Factual Description
bananana	8	Convenient, but short shelf life
strawberry	9	Pretty good
pineapple	6	Tasty, but a hassle
regular apple	-4	Pointless
cantaloupe	99999999999	The reason I live
watermelon	99999999998	Inferior to cantaloupe

Some resources and websites to get you started are below. You are encouraged to be as creative as possible here, pursue some of your personal interest, or relate it to your future career.

- www.census.gov
- http://factbook.gatech.edu/quick-facts/admissions-enrollment/
- http://m.bbref.com/m?p=XXteamsXXCHCXX2016.shtml

For the final part of this project, we are going to see the limitations of Excel and see if we can get MATLAB to help us out. Open up "random_people.xlsx" and take a look at the data. What if we wanted to find the first and last name of everyone who has an email? What about if we wanted to find the ip address of everyone with a particular area code? All of these operations are rather difficult to perform in Excel. They are not impossible, but the formulas are not pretty (they are much prettier in Google Sheets however). Instead, we're going to use MATLAB to solve this problem. However, there is one catch here. You may not use any iteration for these problems. That's right, no for loops or while loops. The `cellfun()` function will be very useful!

Requirements:

- **(20 points)** Read and follow steps in "Excel_Project.pdf" file
- **(20 points)** Find your own data set and complete some unique and relevant math operations and general functions .
    - Only using the sum function and adding two columns for no reason will not receive high marks.
- **(20 points)** Heavily incorporate `vlookup` and conditional formatting in final spreadsheet.
    - How you choose to use those techniques is up to you!
    - Your goal should be to impress us and use those tools in a way that shows how helpful they are.
- **(10 points)** Create at least two graphs representing your results.
    - At least one must be for data of type double.
    - At least one must be for data of type char.
    - For at least one of your graphs, make the same type of plot/graph using MATLAB techniques taught in this course.
- **(30 points)** Open up "random_people.xlsx" and use MATLAB to perform the following operations (without using loops):
    - Find a cell array of first names for those people who have a valid email (i.e. the email column is not blank).
    - Find a cell array of the IP addresses of those who have a homework that starts with the letter "S".
    - Find a cell array of emails for those who have a favorite website that is NOT a .com, and who are female.

Extra Credit:

- **(35 points)** Do one more `cellfun()` operation.
    - Take a look at the Excel file "chipotleOrders.xls". This file contains real Chipotle orders that you will be analyzing.
    - You want to determine what the most popular thing at Chipotle is. You could just determine the most popular item (Barbacoa Burrito, Steak Bowl, etc), but you want to go a step further and determine not only the most popular item, but also the most common combination of toppings for that item. However, the catch is that, like with the previous `cellfun` portion of the project, you MAY NOT use iteration at all to do this. Feel free to use `cellfun` and search around for other functions that may help you in your endeavour.
    - The final output of the function that you write should be a string that says the most popular item (the item that occurs the most) and the list of the most common set of toppings for that item. Note that you are not looking for simply the most common toppings, but rather you are looking for the most popular COMBINATION of toppings for the most popular item.
- **(30 points)** Using Excel Macros, create a "table of contents" sheet with buttons to navigate between the different sheets of your document for part 2. There should be a button for each sheet, and when clicked, it should navigate to that sheet automatically.

- **(35 points)** Redo all of the calculations you did with MATLAB (using `cellfun()`) in Excel using Excel formulas. Make a new column for each calculation.

Hints:

- The `cellfun()` function will be extremely useful when writing the MATLAB function. Combine that with logical indexing and you just saved yourself the trouble of writing loops over and over again.

What to turn in:

- Analysis text file (see first page)
- `Excel_Project.xlsx`
- Excel file with your dataset and analysis
- MATLAB function for `'random_people.xlsx'`

Project Option #3: 3D Animated Clock

Motivation:

In this project, you will use your MATLAB skills to build a fully-functioning 3D clock.

Description:

The function `analogClock()` should be able to make the face of an analog clock and animate the movements of the second, minute and hour hand appropriately. You are not required to have any other functions besides `analogClock()` when you hand in your files, but it is encouraged that you make helper functions to aid your main function.

ABCs

Before you begin writing your `analogClock()` function, you should and must first finish the ABCs for surface plotting and animation. There are 3 ABC files total, one on surface plotting and two on animation. Unlike other ABCs you have had this semester, there is no autograder to check your answers for these. The only way to see if you completed the ABCs correctly is to run the solution code (`.p` files) and visually compare your answers. Ensure that you load the `ABCs_surfacePlotting.mat` file and run the surface plotting ABCs with inputs when testing.

`analogClock()`

After completing the ABCs for surface plotting and animation, you should now feel comfortable making the face of a clock with a second, minute and hour hand. You are encouraged to make a flat disk your clock face and use flat cylinders as your hands (think about how you would use the `sphere()` and `cylinder()` functions to achieve this, and how you would manipulate your `xx`, `yy`, and `zz` values . However, you are in no way required to make your face in this manner. You are allowed (and are highly encouraged) to use the `rotate` function to rotate your hands appropriately. Otherwise, you may use the 3D rotation matrix.

You have also been provided with the `freezeColors()` and `unfreezeColors()` functions. These functions allow you to use multiple colormaps in a single figure. You can call `freezeColors` (without inputs) after you use the `colormap` function.

The file `analogClock_sample.p` has also been provided for you to see an example of what your animation may look like. Try running the `analogClock_sample()` function with an input of `'now'`. An `analogClock()` function file has already been created for you with some comments and simple code to help you get started. You are free to create a new file altogether if you wish.

Requirements:*

- **(30 points)** ABCs
    - **(10/30 points)** ABCs_surfacePlotting

    - **(10/30 points)** ABCs_animation1
    - **(10/30 points)** ABCs_animation2
- **(70 points)** analogClock
    - **(20/80 points)** Clock has a 3D face
    - **(20/80 points)** Clock has three 3D hands (hour, minute, second)
    - **(12/80 points)** Clock hands move in real time
        - The seconds hand should rotate around completely in a minute.
        - The minute hand should rotate around completely in an hour.
        - The hour hand should rotate around completely in 12 hours.
    - **(10/80 points)** Minute and hour hands positions interpolated in
    - **(8/80 points)** Axes turned off

You will receive 0 credit for making a 2D plotted clock!

Extra Credit:

- **(10 points)** Incorporate the date somewhere.
- **(10 points)** Put in 1-12 hour marks and/or other tick marks.
- **(20 points)** Make the clock face more intricate (a grandfather clock, with a swinging pendulum).
- **(20 points)** Ticking noises / hourly clock chimes
- **(20 points)** Change the background based on the time of day.
- **(10 points)** Incorporate a second input which specifies the format of the clock (pocket watch, sports watch, etc.).
- **(10 points)** Add images to the clock face.

Notes:

- You must create the face of a clock with a second, minute and hour hand using surface plotting.
- The clock should work properly like any other clock would.
- You can create additional functions if you want to do more than what the provided function guidelines offer.

Hints:

- Use `colormap()` to change the color of your surfaces.
- Use `freezeColors` after calling `colormap`.
- Consider what shading interp does and whether you need it.
- Think about when you need to `hold on` and `hold off`.
- Don't forget to `pause`.

What to turn in:

- Analysis text file (see first page of PDF)
- `ABCs_surfacePlotting.m`
- `ABCs_animation1.m`
- `ABCs_animation2.m`
- `analogClock.m`
- Any other files that contain code for this project

Project Option #4: 2048 Game GUI

Motivation:

For this project, you will be creating your own MATLAB version of the tile shifting game 2048! You'll be working with GUIs (graphical user interfaces) in MATLAB.

Description:

To get you started, we are providing an optional ABCs to introduce some MATLAB features that may be helpful. There is also a document called Figure Basics in MATLAB which goes along with these ABCs.

We are also providing you with a skeleton function that will guide you through how to build this project. This is just for guidance, so you are free to change this and implement your project in any way. We have also given two helper functions called `boardSlider()` and `tileGenerator()`. These can help you manipulate a backing array of doubles, which can represent the board, by sliding the values based on user input and randomly generate tiles into the array, respectively. You can look at the help documentation for more information on how these helper functions work.

Requirements:

- **(10 points)** Properly set up the figure window with appropriate dimensions, background color, and figure title, and remove the menu and the number title
- **(5 points)** Display the title of the game 2048 in the window
- **(5 points)** Display the score in the figure
- **(10 points)** Create the appropriate number of tiles objects
- **(30 points)** Use arrow key input controls to manipulate backing array
- **(10 points)** Update tiles within the figure to match the array
- **(5 points)** Have different colors for blank tiles vs. a tile with a value
- **(5 points)** Update the score display as the game is played
- **(10 points)** Display a game over page when there are no more available moves
- **(10 points)** Includes a quit button in the figure that will exit the game

Extra Credit:

- **(20 points)** Include a start screen when the game is run. This can include a start button, color settings, game title, etc.
- **(20 points)** More advanced display during game play. Includes color changing titles, displaying gamer tag, displaying previous high score, etc
- **(10 points)** Include a play/pause button. Should prevent the game from being played while the game is paused.
- **(20 points)** Animate the tiles so they move across the board as the game is played.
- **(20 points)** Create your own helper functions
    - **(15/20 points)** `boardSlider`
    - **(5/20 points)** `tileGenerator`

- **(10 points)** Create a more advanced game over display window. This can include a flashing or animated game over display, a return to start screen button, a replay button, etc.

Notes:

- If you use `waitforbuttonpress` and it errors when you close your game window, then you will not have points taken off. However, it is highly encouraged that you write you code so that no errors occur.

Hints:

- You may find the function `waitforbuttonpress` helpful while completing this project
- You may find the figure properties `CurrentCharacter` and `KeyPressFcn` helpful for this project

What to turn in:

- Analysis text file (see first page)
- `play2048.m`
- `boardSlider.m` and `tileGenerator.m` (if you choose to complete them)
- Any other files that contain code for this project

Project Option #5: Build your own website

Motivation:

Have you ever been close to the end of the semester and wondered "what do I need on the final to get a _____ in this class"? In this project, you're going to build a custom website that can help you answer just that question. Instead of the usual MATLAB, we're going to reach outside of our normal toolbox and explore other coding languages – specifically web languages like HTML, CSS, and JavaScript. Don't panic, we're not going to do anything too complicated with them.

Description:

For this project, you will build a website that helps you figure out what score you need on the final to achieve a certain score in a class, and you are going to upload that website to your Prism drive. The Prism drive is the space allocated by Georgia Tech just for you. That's right – every GT student has their own space on the Georgia Tech servers that they can use to host websites (like resume websites, blogs, etc.). For our purposes, we're going to pretend that every class is broken down into 5 sections:

1. Tests (before the final)
2. Homework
3. Quiz
4. Class Participation
5. Extra Credit
6. Final Exam

But every class assigns different weights to these categories. For example, one class might assign 30% to tests while another class might assign 85%.

You will build a user interface that allows the user to enter the percentage assigned to each category, the grade they have for that category, and what grade they want coming out of the final. Then, your website will display what score they need on the final to achieve that score in the class.

Requirements:

- **(20 points)** Website inputs
    - **(10/20 points)** Having inputs for each of the 6 categories above for their associated weight
    - **(10/20 points)** Having inputs for each of the 6 categories above for the user's score in that category
- **(10 points)** Desired grade and button
    - **(5/10 points)** having a user input to enter what score they want in the class
    - **(5/10 points)** for having a button to calculate the score needed
- **(50 points)** Grade calculation
    - The correct score shows when the button is pressed

- **(10 points)** Input validation
    - display an error message if the user enters percentages that don't total to 100%
- **(10 points)** If the page has at least 10 unique lines of CSS styles
- **Note – you will receive no credit if your site is not publicly accessible at your Prism URL!**
    - **Your Prism URL is http://www.prism.gatech.edu/~yourGTUsername/**

Extra Credit:

- **(10 points)** Extra CSS styling: at least 25 different style definitions
- **(20 points)** the user can enter a comma separated lists of scores into any category, and the score for that category is calculated as the average of the comma separated list
- **(70 points)** Allow the user to search for a class and dynamically populate category weights (see the last page)

Notes:

- The extra credit user input has no "weight" -- the value entered into "score" is added onto the final grade

Hints:

- HTML and CSS
    - Here are a few slideshows for introductions to HTML and CSS.
    - If you want more information on HTML and CSS, there are lots of great resources online. Probably more than you could go through in a year. Google is your friend here.
        - HTML inputs
        - HTML buttons
- JavaScript
    - There are also lots of great tutorials online regarding JavaScript. But since our project is simple, you will probably find a lot of success in just searching your question. Examples:
        - "How to handle a button click in JavaScript"
        - "How to get text from input in JavaScript"
- Uploading your site to Prism
    - To upload your website to your Prism drive, you can follow the walkthrough put together by OIT.
- Calculating the grade
    - If I had two categories in a class, test and homework, I could calculate the grade I need on the final with the following code:

```
test_raw = test_score * (test_weight/100)
hw_raw = hw_score * (hw_weight/100)
score_needed = (desired_score - extra_credit – test_raw – hw_raw) /
      (final_weight/100)
```

- Here's an example of a barebones solution (don't copy this!)

- http://www.cs1371.gatech.edu/website/

What to turn in:

- Upload the following files to Canvas:
  - Analysis text file (see first page of this PDF)
  - All code (HTML, JavaScript, and CSS) files
- Upload your website to your **Prism URL**
  - **http://www.prism.gatech.edu/~yourGTUsername/**

Getting Started on the Extra Credit

For the last part of the extra credit, instead of having the user input the weight of each category, you should allow the user to choose a class from a list, and then you will go get the category weights for that class. To do this, we have built a mock API which returns the weights of classes. In the context that we're dealing with, an API is just an interface that allows you interact with data on a website.

Our API returns the data associated with the weights for a given class. You can access this API by using a specific URL like this:

http://cs1371.gatech.edu/getClassInfo/?class=className

Where "className" is the class you want the data for. Here are some example URLS, you can try them out in your browser or just follow the hyperlink:

- http://cs1371.gatech.edu/getClassInfo/?class=CS1371
- http://cs1371.gatech.edu/getClassInfo/?class=MATH1501

If you try out those URLs, you'll notice that the result is not a nicely formatted HTML page like you're used to. Instead, you see a bunch of data formatted as a JSON string. This is much easier for programmers (such as yourself) to work with. Don't let JSON scare you, it is just one of the many ways to format data. Luckily for us, JSON is very easy to use in JavaScript.

The data returned from our API will have each category and the corresponding weight for that category. It will also have a "success" field that has a value of true if the class was found and false otherwise.

To earn this extra credit, you should present a list of classes to the user that they can choose from. When they select one (you can have another button for selection if you want), then your website should go get the data from the API shown above, and fill in the weights in the appropriate boxes. Here's a list of classes that our API will display fake data for: CS1371, AE1350, AE1601, AE1770, CHEM1211, CHEM1212, CHEM1315, CHEM2311, ECON1101, ECON2100, ECON2101, ENGL1101, ENGL1102, MATH1111, MATH1113, MATH1501, MATH1502, PHYS2211, PHYS2212, PHYS2213, PHIL2010, PHIL2025, PSYC2005, PSYC2020, PSYC2103, NRE2110, NRE2698, CS1331, CS1332, ECE1010, ECE2020, ECE2026.

Note that this type of request is called a "GET" request because we are passing the necessary data inside of the URL. Feel free to search around for how to make a GET request in JavaScript