Final Project Description and Requirements

As part of the course EGR 115, you are required to produce a MATLAB program of your own design and construction that provides a solution or capability for a topic of your choosing. The purpose of the project is for you to demonstrate your skills in programming.

Goal: Implement/use as many of these concepts as you can with reasonable purpose for your program - in a correct, reasoned, and logical manner.

LIST OF CONCEPTS

Modular design and implementation
- Algorithm documented inline as comments
- Library functions
- Programmer-defined functions
  - Be sure that they use parameters and that they return values

Data structures
- Arrays
  - Use vectors and/or matrices (other than for strings)
- Strings
  - Creation / concatenation: array building, sprintf()
  - Truncation; parsing; searching
- Secondary storage (file I/O)
  - Read data from files into your program's data structures
  - Write data from your data structures to disk

Program control
- Conditionals: IF, SWITCH
- Loops: FOR, WHILE

Data processing
- Running totals/products
- Array-building
- Sorting
- Searching
- Random numbers

User Interface
- Command window
  - Input (strings and numbers)
  - Formatted output (format strings, format modifiers, escape sequences)
- Dialog boxes

Plotting
- Cartesian coordinates and/or polar coordinates
- Formatting and labeling

Other (not required; possibly extra credit)
- Custom GUI
- Modification of figure properties
- Multimedia (audio/video)
- Toolboxes

REQUIREMENTS.

Besides implementing as many concepts as possible, your program must also satisfy these requirements:

1) Main Program / Function

   The file containing the starting point of the program (either the main program, or starting function in the case of GUIs) must be named "Final_Project_Main_LastName.m"

2) No error messages. Red error messages put out by MATLAB must be eliminated.

3) No serious logic flaws.

4) Single Interface.

   Your program should use only one interface: CLI (command-line interface – aka, command window) or GUI (Graphical User Interface: your own or using standard dialogs). If you are using a GUI and have text output, use sprintf() to create nice-looking output.

5) Validation.

   Make sure that all user keyboard inputs are validated – that the user provides the right data type (string, number), form (e.g. length of string) and magnitude. If the validation fails, have the user re-enter. Do not proceed until the user provides good data.

6) Allow the user to contribute.

   The program must be written so that the user can add/change/delete data items through the program interface. The first module to be written is usually that which allows the user to add to the data file. For example, an expense tracker would have a set of expense categories that the user can modify through the program. Or in a game environment your programs allows the user to save the state of the program and then return to that state at another time (e.g. save a game and load it later).

DESIRABLE TRAITS

1) Reasonable use.

   Using a feature from the concept list must make sense for your project. Also, using a feature just to fulfill the requirement when a better option is available won't count. You will not receive credit for "fluff".

2) Strive for efficiency.

   Always look for the most efficient algorithm. Many times this require a little thought to determine the pattern or math required.

   Make good use of library functions; feel free to explore and use the ones not covered in the lectures, if that makes your code more efficient.

   Avoid extraneous SWITCH/IF statements

   Create programmer-defined functions if that block of code will be called again and again in your main program. If you are writing multiple functions that perform essentially the same task, you need to reduce to a single function with a parameter.

3) Nicely-formatted.

Algorithm documented inline as comments are highly advised.

Indent all code neatly and uniformly (use Ctrl-A-I).

Properly use whitespace, within lines and/or between lines, to enhance readability.

4) User-friendly.

Make the program easy for the user. Never ask for input without providing the type and magnitude expected. Visual and auditory feedback is important for the user. Sometimes a picture, graph, chart, or beep is more valuable than numeric data.

5) Generality; avoid excessive hard coding.

Make your program flexible. Your program should be able to handle reasonable variations in the data set in terms of size and type. If you do have excessive hard coded values, use a date file to store this information.

6) Be cautious with dangerous techniques.

There are some techniques that are useful in experienced hands (such as a recursive function – a function which calls itself) and some that are bad form and should never be used (parental function calls – calls to the function which originally called the current function). Use extra caution if you plan to play with these.

SUBMISSION

You should begin designing your project as soon as possible.

Phase 1

A written proposal on what you plan to do for the final project is due on Thursday, 3/19/2015, at our regular lab time. The proposal needs to include a detailed description of the problem, basic plans to tackle it, the expected inputs and outputs, and user-friendly features. For examples, if you are to build a game, the rules and features of the game need to be fully explained in the proposal. Or if you are to solve a physics problem, the equations, assumption, and solutions need to be complete and easy to read.

I expect communications between you and me, before and after the submission of the proposal.

I will review your idea and provide you with my opinion of its potential and/or "level of demand" - how hard/time-consuming I think it will be for the typical student.

Phase 2

Your final project is due at 11:59pm on Thursday, 4/23/2015 (the last day of school). Your final submission has two parts, a project report and a zipped folder that contains all scripts and data files. Submit both to the Blackboard.

The report is a full detailed description of your work. It may include, but not limited to, the following sections, background & introduction, illustrations & equations (if any), the algorithm, the program set-up, difficulties and solutions, testing cases and the outcomes. You are encouraged to take screen shots and paste them to your report to better illustrate/demonstrate your product.

There is 20% per day late penalty.

Appendix: Final Project Ideas *Credit to Matthew Kindy

Some general areas and examples of their implementation are listed below. Please do not consider these your only options - your project is limited only by your imagination and time! Topics that are no longer accepted are crossed out.

Games
Diablo II for Allegro
Keyboard DDR
Drum Hero (like Guitar Hero for drums)
Super Mario Brothers
Number guessing (banned)
MasterMind
Monopoly
Blackjack (be careful – this one is commonly "under-developed")
Tic Tac Toe (banned)
Deal or No Deal
Connect-4
Checkers
Pong

Databases (probably the easiest way to use everything in the course)
Store-front
Hotel check-in
Student attendance using mag-card reader
Retail check-out using bar-code scanner
Sports teams (bowling, soccer, football, etc)
Tournament generation (banned)
Automobile parts
Music database
DVD database

Hardware control & Data collection
Serial port transmission and reception with GUI (graphical user interface)
Program to collect and report temperature readings
Simulators
Rocket launch simulator (banned: too similar to EGR101 project)
Airplane simulator
Calculators
Physics problem solver (banned)
Projectile calculator (banned: same as physics problem solver)
Planet computer (banned)
Aircraft attribute calculator (banned)
Unit conversion (Max 75 points unless user can provide new dimensions and units)
Sniper rifle mil-dot calculator
Bezier curve fit
EGR101 projects (banned)
Blood Alcohol Calculator (banned)
Multimedia
MP3 player

Movie player
Keyboard simulator
Audio tuner
Musical Synthesizer
Quiz/Test Generators (Be careful of excessive hardcoding. The user must be able to add questions to the database)
Student Course Planner, track courses taken toward major and minor degrees

Other ideas
Genealogy (family tree) program
Advanced physics demonstration (combo of fluids/statics/dynamics)
Heat transfer computational demo
Fluids demo program with graphics and videos
External data processor (save an external data source as a file and process that data, e.g. pull data from website and produce searchable database)
Store's inventory-control system using a bar-code scanner
Job scheduler – schedule employee shifts, handling days off, vacations, avoiding overtime, avoiding "back-to-back" shifts, etc.

Reminder: Plagiarism and other forms of cheating on the final project will result in failure in the course and possible referral for expulsion. Better to turn in nothing than to get caught cheating.