**SE 185 Final Project**

This is a **group assignment** and the group size should be **2-4 students**. Group members do not have to be from the same lab section, but it is recommended as it’ll be easier to find work time.

You must select your team members and the final project to submit it to the “Final Project Selection” assignment on Canvas by Monday, April 24th at 11:59 PM (Note that this is a group assignment and one submission is required per group).

The submission deadline for this assignment is Friday, May 5th at 11:59 PM. Submit your files to the “Final Project” assignment on Canvas. (Note that this is a group assignment so only one team member needs to submit). You must use the template provided with this document (FinalProjectTemplate.c). This will just be a submission of your code. No lab report is required.

You can demo your project to your TA during your lab section or your lab TA’s office hours on the following week: **The week starting May 1st, 2023 (Dead Week).** No demos will be accepted after Friday, May 5th at 11:59 PM.

**Objectives**

The main objective of the final project is to teach you how to put together all the class material that you have learned so far in order to create a larger C program that accomplishes a specific task. The goal is to show how programs are designed in the real world.

**Project Selection**

There are two projects to choose from. They are of approximately the same difficulty. **Your grade will not depend on the project that you pick**, as long as you implement the entire project.

If you don’t like any of the provided projects (1 or 2), then you also have the option of **specifying your own project.** If there was something that you always wanted to do with C, now is your chance. However, your proposed project MUST be of appropriate complexity. **Write out your proposal and submit it into the “Final Project Selection” assignment on Canvas.** We will reach out to your team about whether your project is accepted. If you would like to start early, you can talk to one of the TAs early to get verbal acceptance (you must still have your project description written out).

**Notes on the Two Provided Projects**

If you choose to implement one of these projects, you must follow all the details listed in the project description. Your implementation, at minimum, should satisfy all the implementation features and constraints as specified in this document. This means that if you do not implement a certain feature or reduce the complexity of the project you will lose points as indicated in the grading rubric. It would be helpful to discuss any changes with your TA.

Also, there will be ZERO credit for any design step in the grading rubric that does not work or produces the wrong output or is not of the given specifications.

**General Development Tips**

* Start early!
  + This gives you more time to get help if needed and verify that everything is working correctly.
* Work together!
  + Try to find a time that all your team members are available. Also, it will be helpful if your team can fix two meeting times per week to work on the project.
* Get help from your team members!
  + This is a team project. Even if you divide up the work, everyone is responsible for each part of the project. Utilize each other’s expertise and don’t be afraid to ask for help.
* Get help from us!
  + Your TAs are here to help you. We can’t do the entire project for you, but if you are spending hours on a single bug you should ask for help.
* Break up the project!
  + Try to find a job for each person. This doesn’t mean that it is all they have to do or that they can’t get help. But this makes the workload fairer. (Remember that your team must specify the participation percentage of each team as shown in the project template.)
* Work iteratively!
  + Don’t try to complete the entire project without even testing your code. Try to implement a small part of a requirement, test it works. Then implement more. This will help you catch bugs sooner and reduce your frustration.
* Comment your code!
  + You're not the only one reading your code. Make it clear what you are doing and why you are doing it. If you are working on a team project in Google or Facebook, you will find that they always heavily comment on the code so that all team members can understand every chunk of the project code.
* Use your resources!
  + Look things up if you don’t know. The internet has a ton of available information that can help. Also, previous labs have similar segments that could be used as a starting point.

**Project #1: Typing Game**

**Scenario:** You are to create a typing game that can be played through the Cygwin terminal. Words will appear in a box and the player will need to type the words before they reach the bottom of the box or the game will end.

Sample Pictures on Canvas

Code Requirements:

* Use of the provided text file (on Canvas) to produce the words that are used in the game.
  + The words that are used in the game should be randomly chosen and different for each instance of the game.
* Use of a 2D array for the playing board to keep track of the position of the words.
* Have the computer randomly choose the starting position of each word
  + It should pick its x position; the y position should always be at the top.
  + The word should always be contained in the board, for instance it should always have enough space to its right for its length.
* Prompt the user if they are ready to start at the beginning.
* Give them the option to add words to the text file you are using. The changes should persist between instances of the program.
* Have a way to check if the game has ended (i.e. if a word has reached the bottom of the board)
* The player should be able to type any word they want, not just the one that is closest to the bottom.
* The word must disappear from the next board and no longer affect the game state only after the user has typed the word correctly.
* The time of the player is recorded and displayed at the end.
* The words DON’T need to be generated without user input.
  + It would be very difficult to have words appear while the user can be typing in the console. So we will update and reprint the entire board after the user gives input.
  + The longer a user takes to type in their response the more words spawn. So if a new word is created every second then if the user takes 3 seconds to enter their response, 3 words would be created after the user presses enter. Also every word on the board would move down 3 spaces.
* The rate at which words appear should gradually increase as the game goes on.

Development Tips:

* You will need to read in all the words in the given text file (on Canvas), start with that and verify it works properly.
* Then create three functions:
  + Add a given word at a random x value to the top of the board.
    - Called a number of times when the user presses enter based on the time since they last pressed enter.
  + Move every word down one row on the board.
    - Called a number of times when the user presses enter based on the time since they last pressed enter.
  + Delete a given word from the board
    - Called when a user types the word in.
* Initially just try to get it to run so that it adds a word and moves every other word down when you press enter.
* Then make it so you can type in a word to delete that word.
* Lastly create some kind of timer to create more words the longer the user waits to type.
  + In previous labs we have done similar things.
* This is not something that will be made easier with ncurses.

Rubric:

* (20pts) Ability for the user to add words to the text file being used.
* (30pts) Randomly choose and place the correct number of words each time the user types in a response.
* (15pts) Correctly move all words down the appropriate number of lines each time the user enters a response.
* (30pts) Correctly deleting the intended word after the user types it.
* (5pts) Comments in your code
* (100pts) Total Possible

**Project #2: Memory Game**

**Scenario:** You are to create a memory game that can be played through the Cygwin terminal. Symbols will appear on the console and one will be circled at a time. Then the user will have to enter the order in which the symbols were circled.

Sample Pictures on Canvas

Code Requirements:

* Prompt the user if they are ready to start at the beginning.
* Have the computer randomly choose which symbol is circled.
  + It will also need to keep track of this so it can check it against the user input.
* The program must check for invalid input and respond accordingly, i.e., tell the user it was invalid and prompt them again.
* The total score of the user (how many correct guesses they made) is collected and displayed to the user after each guess.
* The symbols should be unique and semi complex (see the sample output)
  + No single character symbols
  + Must use some kind of loop to generate at least one of the symbols.
* Include a highscore table
  + It should prompt the user to enter his/her initials if their time is better than one of the current top ten scores.
  + It should then print the updated table of high scores.
* It is okay if the user theoretically could just scroll up in the console to check what was played. We will assume no cheating.
* The maximum score you need to account for is 99, any score above that can be inputted to the high score table as 99.

Development Tips:

* Highscore table
  + This will most likely be the most challenging piece of this project, start it early!
  + This should be saved between instances of the game. You will need to write to a text file to achieve this.
  + Then you’ll need to read from that text file to print out.
  + You may need to create a unique pattern that you can store data in a text file.
    - You control the way you write to the text file and the way you read from it!
    - An example of this is provided, but you can come up with your own format.
  + At the end of the game you will need to check the players score against all the high score values to see if the player got a high score.
* Game State
  + Use previous labs as examples where we made games similar to this.
  + Have functions for all the big steps, like printing out the symbols, to make your code more readable.
  + Make sure your arrays are big enough to hold a sufficient number of rounds.
  + When developing strive for simpler code.

Rubric:

* (5pts) Give proper prompts to the user.
* (10pts) The unique design of the symbols.
* (15pts) Randomly choose and properly select the symbols a correct number of times for the round.
* (15pts) Accept user input and verify it is the correct sequence of symbols.
* (5pts) Account for invalid input from the user.
* (20pts) Correctly reading in and displaying the high score table from a text file.
* (25pts) Properly updating the high score table at the end of each game when the score requires it.
* (5pts) Comments in your code
* (100pts) Total Possible

**Project #3: Propose your own project**

Create Your Own Program and Scenario. Please contact your instructor (Maruf) immediately to propose your own project. You project must be approved by your Instructor (Maruf) before working on it.

**Scenario**: Propose your own scenario.

**Must meet all:**

* Pointers with dynamic memory allocation
* Use of Structs
* Use of user-defined functions

**Choose 1:**

* File I/O
  + Read in a file
  + Output a file
* File hierarchy
  + Minimum 2 header files
* Use ncurses
  + Use mvprintw to print to the screen

**Rubric:**

* (60pts) Required Section
* (35pts) Choose 1
* (5pts) Comments
* (100pts) Total Possible