The Final will be graded on a total of 100 points.

There are four programming exercises each worth a total of 25 points. Additionally, there will be a quiz available worth 50 points. The quiz is open book, open notes, and open Internet but you will not be allowed to discuss your answers with others.

The exam questions are to be completed on your own – any plagiarism will be considered a violation of the Academic Integrity Policy that is outlined in the course syllabus.

For each of the programming assignments that you turn in, be sure to include well-placed comments indicating what your code does. Be sure to include user directions so that it is clear what the user is required to do in order to get your program to function.

Program 1: Password Protected

Create a program that askes the user to enter a password. The program should then utilize a function to verify the password meets the criteria below. The program should then display a message indicating if the password was valid or invalid.

The password should:

Be at least 12 characters in length
Contain at least one uppercase character
Contain at least one lowercase character
Contain at least one number
Contain at least one special character (!, @, #, $, %, ^, &, *)

Program 2: Total GPA Calculator

Create an application that asks how many total classes the student has taken. For each, the program should use programmer defined methods to input the following information: the course number, the name, the letter grade, and the total credit hours. The program should then utilize programmer defined functions to calculate and display the quality points for each class as well as the semesters total GPA and display it to the user. If the GPA is higher than a 4.00 or lower than a 0.00 display an error message, otherwise display the corresponding letter: A = 4.00, A- = 3.67, B+ = 3.33, B = 3.00, B- = 2.67, C+ = 2.33, C = 2.00, C- = 1.67, D+ = 1.33, D = 1.00, F = 0.00.

Program 3: Bank At Home

Create a program that provides access to a bank account with a Saving and Checking account balance. The amount in both accounts should be displayed and updated after each transaction. Users should NOT be able to change the balance directly for either account and instead be done through the use of functions. Users should have the option of depositing, withdrawing, or transferring funds. Users need to be able to specify any dollar amount. Users should NOT be able to withdraw or transfer funds great than what is available in their account.

Program 4: Computer Guessing Game

Write a program that allows the user to play a guessing game with the computer. The user should pick a number at random. They should keep this in their head and not tell the computer. The user should then be asked to enter a starting range value and ending range value (e.g., 1 and 100). The computer should inform the player that they will guess the number the player is thinking of in a number of guesses equal to the square root of the difference of the highest value minus the lowest value. (e.g., Square Root(100 – 1) = 10). After each guess, the player is responsible for telling the computer if the value the computer "guessed" was too high, too low, or correct. If correct, the computer declares itself the winner and ends the game. If too high, the highest value should be set to 1 minus the last guess. If too low, the lowest value should be set to 1 plus the last guess. The next guess should be determined by a function that returns median between the highest and the lowest value. Before each guess, the computer should announce what guess it is on, how many guesses it has left, and the value that it is guessing. If, at any time, the high value is less than the low value, or the low value is greater than the high value, the computer should tell the player they are cheating. If the computer runs out of guesses before getting the correct answer, it should declare the player the winner and end the game.