COMP 1405Z – Fall 2022
Introduction to Computer Science I
Practice Midterm

Practice Midterm Details

The real midterm will cover all topics up to and including functions. **You should ensure you understand functions and are comfortable with writing them before the midterm. Each question on the midterm will involve creating at least one function.**

Part 1

Write the code for this part in the part1.py file. Write a Python guessing game. Generate a random number between 1-100 and have the user guess numbers until they are correct. Any time the user guesses wrong, the program should output a hint indicating ‘guess higher’ or ‘guess lower’. When the user guesses correctly, print out a message indicating that they have won the game. You should run and test your code to verify it works correctly.

Part 2

Write the code for this part in the part2.py file. Write a function that asks a user to enter two numbers (X and Y) and returns the greatest common divisor of X and Y. The greatest common divisor is the largest number that divides evenly into both the given numbers. For example, the greatest common divisor for 20 and 24 is 4. If either of the input arguments are less than 1, your function should return -1. Provide code within the file to test your function.

Part 3

Write the code for this part in the part3.py file. Write a function called sumdivisors that takes a single integer input argument N. The function must return the sum of all positive integers that evenly divide into N. That is, the sum of every number X where N divided by X produces no remainder. For example, if N is 9, the function would return 13 (1 + 3 + 9). If N was 18, the function would return 39 (1 + 2 + 3 + 6 + 9 + 18). You can run the part3-tester.py file to test your solution. This will load your solution and execute the function with several input values. It will summarize the number of correct output values your function produced.

Part 4

Write the code for this part in the part4.py file. For this problem, you are given several files (volumes0.txt – volumes4.txt) in the ‘exam’ directory that contain the dimensions of boxes. You should assume that all values for this problem are floats. The structure of each of these files will contain three lines of information for each box, representing the length, width, and height of the box, respectively. So, the first/second/third line of the file contain the length/width/height of box #1, the fourth/fifth/sixth lines contain the length/width/height of box #2, etc.

Write a function called largestvolume that takes a single string input argument representing a filename. This function must return the largest volume of any box from the given file. Note that the volume of a box can be computed as `length x width x height`. You can assume that any filename you are given will represent a file that follows the specified structure. You can run the part4-tester.py file to test your solution. This will load your solution and execute the function with the provided files as input. It will summarize the number of correct output values your function produced.