CSCI 135 Programming Midterm Exam
Fundamentals of Computer Science I
Fall 2021

This part of the exam is like a mini-programming assignment. You will create three (or four if you do the bonus program) programs, compile them, and debug as necessary. This part of the exam is open book and open course website. You may use code from the course web site or from your past assignments. When you are done, submit all your source code files to the Moodle Midterm dropbox. Please ***double check you have submitted the required Python source code file***.

You will have until the end of lab at 6:00 PM. No communication with any non-staff members is allowed. This includes all forms of real-world and electronic communication. It also means no electronic devices other than the computer you are using.

Grading.

Overview. There are three short programs to write in this part of the exam. The last of the three has the opportunity for bonus points if you choose to do it. You are not required to. Programs two and three are worth 15 points each, and program one is worth 20 points. All programs should be turned in to the Exam 1 dropbox on Moodle.

Problem One: Guess my number. Guess.py (20 points). Randomly generate a number between 1-100. Prompt the user to guess the number. If their guess is higher, print "Too High! If their guess is lower print "Too Low!". Repeat this until they guess the correct number, when it should print

"Correct! Your number was X and it took you Y tries to find it.".

Black is printed, red is user input, the generated number in this example is 22.

```
Enter your guess: 66
Too High!
Enter your guess: 20
Too Low!
Enter your guess: 40
Too High!
Enter your guess: 30
Too High!
Enter your guess: 22
Correct! Your number was 22 and it took your 5 tries to find it.
```

Problem Two: Seven, Eleven, or Doubles. SED.py (15 points).

Write a program that simulates rolling two dice 1,000 times and displays the number of occurrences of when the dice added together totals 7, when the dice added together totals 11, or when the two dice are the same. Format this output however you like as long as numbers are labeled. Name this program Dice.py.

Problem Three: Zombie Distance. ZDistance.py (15 points)

Write a program that creates a 6x6 matrix of the number 0. Then it should generate random (x, y) coordinates for a player and print them on the grid represented by a 1. Print the matrix so it shows like this (with 1 being randomly placed)

```
000000
000000
000000
000000
000000
001000
```

Then print the distance your point is from the last point, (5,5). The distance formula is

$$d=\sqrt{((x_2-x_1)^2+(y_2-y_1)^2)}$$

To take the square root, import math and use the following example y = math.sqrt(x) where x is what you want to take the sqrt of, and y is the variable it gets assigned to.

Your final output should look like the following. (Next Page)

```
000000
010000
000000
000000
000000
000000
The distance you are from the end is  5.656854249492381
>>>
==== RESTART: C:\Users\brent\AppData\Local\Programs\Pyth
000000
000000
000000
000010
000000
000000
The distance you are from the end is  2.23606797749979
>>>
==== RESTART: C:\Users\brent\AppData\Local\Programs\Pyth
000000
000000
000000
000000
000000
000001
The distance you are from the end is  0.0
>>>
==== RESTART: C:\Users\brent\AppData\Local\Programs\Pyth
000000
000000
000001
000000
000000
000000
The distance you are from the end is  3.0
>>>
```

Extra Credit: Zombie Distance 2. ZDistance2.py (+5 points)

You will need to submit a separate program that adds onto your code from ZDistance.py. Be careful not to overwrite your old program.

Add a zombie with randomly generated coordinates represented by the number 2 on the grid.

If you and the zombie are on the same spot print “You Died.”

If you and the zombie are not on the same spot print “The distance you are from the zombie is X” with X being the distance.