**Name: Session:**

**Programming I**

**Lab Exercise 9.25.2024**

**When you have completed these programs, submit your documented source code.**

1. Write a function multAdd that takes three parameters, a, b and cand computes a ∗ b + c and returns the result. Write some code to test your function.
2. Write a new function called yikes that has one argument and uses the multAdd function to calculate the following:



There are two different ways to raise e to a power-check out the math module documentation. Be sure to return the result! Try x=5 as a test; your answer should look like:

yikes(5) is 1.0303150673

1. Use the multAdd function that you previously wrote to make the following two calculations:



Hint: If you are unfamiliar with the notation ⎡ ⎤ this represents the ceiling of a number. The ceiling of some float x means that we always “round up” x. For example, ⎡2.1⎤ = ⎡2.9⎤ = 3.0. Look at the math module documentation for a way to do this!

If everything is working correctly, your output should look like:

*angle\_test* is:

1.06066017178

*ceiling\_test* is:

17.55397881653

1. Write a method rand\_divis\_3 that takes no parameters, generates and prints a random number, and finally returns True if the randomly generated number is divisible by 3, and False otherwise. For this method we’ll use a new module, the random module. At the top of your code, underneath import math, add the line import random. We’ll use this module to generate a random integer using the function randint, which works as follows:

random.randint(lo, hi)

where lo and hi are integers that tell the code the range in which to generate a random integer (this range is **inclusive**). 0 to 100 is probably a decent range.

1. Write a function rollDice that takes in 2 parameters -the number of sides of the die, and the number of dice to roll -and generates random roll values for each die rolled. Print out each roll and then return the string “That’s all!” An example output:

>>>roll\_dice(6, 3)

4

1

6

That’s all!

1. Given any initial natural number, consider the sequence of numbers generated by repeatedly following the rule:

* divide by two if the number is even or
* multiply by 3 and add 1 if the number is odd.

The Collatz conjecture states that this sequence always terminates at 1. For example, the sequence generated by 23 is:

23, 70, 35, 106, 53, 160, 80, 40, 20, 10, 5, 16, 8, 4, 2, 1

Write a Python program that generates this sequence. I suggest that your code prints out the sequence of numbers generated by this rule. Run this program for n = 217. What is the largest number in the sequence generated by this starting value?

To test your code, starting at n = 23 generates a sequence with a maximum value of 160.

1. (Extra Credit) Write a function report card where the user can enter each of his grades, after which the program prints out a report card with GPA. Remember to ask the user how many classes he took (think -why would we need to ask this?) Example output is below.

How many classes did you take? 4

What was the name of this class? Computer Programming

What was your grade? 94...

REPORT CARD:

Computer Programming - 94 (A)

Calculus - 88 (B)

French - 76 (C)

History - 91 (A)

Overall GPA = 3.25

Hints:

Each class is worth 3 credits.

Use the following grade scale:

90 – 100 = A = 4.0 grade points

80 – 89 = B = 3.0 grade points

70 – 79 = C = 2.0 grade points

60 – 69 = D = 1.0 grade points

0 - 59 = F = 0.0 grade points