Introduction to Java

CS9053 Section I2

Thursdays 6 PM – 8:30 PM

Prof. Dean Christakos

Feb 2nd, 2023

Due: Feb 10th, 2023 11:59 PM

**Assignment 2**

**Part I – Procedures/Functions**

Looking back on Assignment 1, Part III, we had the WireCapacitance. The capacitance of a pair of parallel wires is given by

and we want to file the difference in capacitance as we move from d0 to d1

Instead of implementing it in the main method, we have created a function called calculateWireCapacitance. It looks like this:

**public** **static** **double** calculateWireCapacitance () {

**return** -1;

}

The function now has no arguments, and the return value, -1, is just a placeholder.

|  |  |  |
| --- | --- | --- |
| **Variable** | **Meaning** | **Value** |
| ε | Absolute permittivity (F/m) |  |
| d0 | Initial distance between wires (m) | .01 |
| d1 | Final distance between wires (m) | .005 |
| *l* | Wire length (m) | .5 |
| a | Radius of wires (m) | .002053 (12AWG) |

Using the above explanation for the variables, modify calculateWireCapacitance so that it takes arguments for l, a,d0, and d1 and returns the change in capacitance, Cd

**Part II**

1. **Boxing weight class.** Olympic style amateur boxing weight divisions for men are:

|  |  |
| --- | --- |
| **Weight division** | **weight** |
| light flyweight | 108 pounds or less |
| flyweight | 115 pounds |
| bantamweight | 123 pounds |
| lightweight | 132 pounds |
| light welterweight | 141 pounds |
| welterweight | 152 pounds |
| middleweight | 165 pounds |
| Light heavyweight | 178 pounds |
| Heavyweight | 201 pounds |
| Super heavyweight | Over 201 pounds |

In WeightClass.java, using if-else statements, take a weight as a float argument and return a String indicating the weight class.

**Part III**

1. In the class DumbPasswords, we will use loops to generate Strings.

The method printDumbPasswords takes two arguments, m and n.

The format of a dumb password is as follows: number-number-letter-letter-number

Character 1: a digit from **1** to **m**.

Character 2: a digit from **1** to **m**.

Character 3: a small letter from the first **n** letters of the alphabet.

Character 4: a small letter from the first **n** letters of the alphabet.

Character 5: a digit from **1** to **m, greater than the first 2 digits**.

printDumbPasswords should print out all the dumb passwords in alphabetical order, separated by a space.

So printDumbPasswords(3, 1) should output

11aa2 11aa3 12aa3 21aa3 22aa3

1. Take this sum:

Figure out, for z = 0 to 15, how many iterations of k it takes for the sum to estimate to within .001. You should print out something like this:

e^0 is 1.0

result for <x> iterations: 1.0

it requires <x> iterations to estimate e^0 within .001

e^1 is 2.718281828459045

result for <x> iterations: 2.7180555555555554

it requires <x> iterations to estimate e^1 within .001

e^2 is 7.3890560989306495

result for <x> iterations: 7.3887125220458545

it requires <x> iterations to estimate e^2 within .001

e^3 is 20.085536923187664

result for <x> iterations: 20.08521256087662

it requires <x> iterations to estimate e^3 within .001

where <x> is the number of iterations.

**Part IV: Arrays**

1. In CopyShift.java, there is the array sourceArray with 50 random integers from 0 to 99. Copy the contents of sourceArray into destArray, but shifted n spaces to the right. For example, if sourceArray has the contents [1,5,6,7,9] and we want to shift it 3 spaces over, destArray should have the contents [6,7,9,1,5].

It should work for arbitrarily large shift values.

1. In FindMinimumLength.java, there is a variable intArray and a goal value goal. Return the minimum length subarray whose sum is greater than or equal to goal. If there aren’t enough values in the intArray to reach goal, return 0.

Examples:

intArray = [1, 3, 4, 2, 1]

goal = 9

minimumLength = 3 (the subarray 3,4,2)

intArray = [4, 8, 6,5,8]

goal = 3

minimumLength = 1

intArray = [1,0,1,1]

goal = 5

minimumLength = 0

1. In TwoDimensionalArray.java, we have a two dimensional array called twoDimArray and two one-dimensional arrays, arrayOne and arrayTwo. You will copy the CONTENTS of arrayOne into twoDimArray[0] and the CONTENTS of arrayTwo into twoDimArray[1]. Use two nested loops and index 0 of the first dimension of twoDimArray means choosing arrayOne as the array to copy and index 1 of the first dimension means choosing arrayTwo.