**PROGRAMMING ASSIGNMENT 2**

Write your responses to parts 2 and 4 in the ‘Programming Assignment Documentation Template’ found in ManageBac. Make sure to include your name and the programming assignment number. Your code can be uploaded alongside the accompanying word document.

PART 1: PROBLEM STATEMENT

I run a company that processes a lot of user data using a lot of searches. In this case, it is helpful to be able to run binary searches on my data as they’re a lot faster. Unfortunately, our data hasn’t been kept up with well and is unsorted, so we need to find the quickest way to sort our data for all of the binary searches we’d like to do on it.

I want you to trial three different sorting algorithms. Two, we covered in class. These are the selection sort and the bubble sort. I’d also like for you to implement the ‘Gnome Sort’, also known as the ‘Stupid Sort’. We’ll analyze the performance of each of these in Part 4.

PART 2: BRAINSTORMING THE ALGORITHM

Before you jump into writing any code, jot down your thought process. Then, write out an algorithm in plain language for the Gnome Sort algorithm, what you intend to do with the code, be descriptive. Do this step in the ‘Programming Assignment Documentation Template’ found in ManageBac.

PART 3: WRITE THE PROGRAM

Write your code in a replit project, or VS Code .py file, whichever is easiest. You will implement three procedures, one for selection sort, one for bubble sort, and one for gnome sort.

You will then modify each of these three methods to count how many times the loops run, to help assess which one is more efficient. We will look at where to implement these counters in class.

We will generate three random arrays containing ten numbers each to test our procedures with. The code to generate an array of 10 random numbers is below.

arr = [0] \* 10

for i in range(len(arr)):

arr[i] = randint(0, 100)

arr = bubbleSort(arr)

Each time elements are moved around in the array, print out the newly changed array to visualize the changes made to it.

Sample output for the bubble sort is provided below.

A screenshot of a computer screen

Description automatically generated

PART 4: REFLECTION

Answer the following questions in the ‘Programming Assignment Documentation Template’ found in ManageBac.

1. How many iterations did each type of sort take on average? If there was any variation, record the highest and lowest number.
2. On average, how many times was the array for each sorting algorithm printed out?
3. What does the previous answer mean, as in, what is the impact of the number of times the array gets printed out?
4. Is there a tradeoff between simplicity of the algorithm and the number of times that your program printed out the array?
5. Overall, which of these algorithms do you think is the best? Back up why you picked the one that you picked.