Lab 9 List

Please finish the following program design and submit via the platform before Nov. 28th and the source code should be saved as the name StudentID_Name_9_1.py, such as 2023333519001_王澜洁_9_1.py, 2023333519001_王澜洁_9_2.py, … etc. and then compress them with the name StudentID_Name_9.rar (or StudentID_Name_9.zip) such as 2023333519001_王澜洁_9.rar (or 2023333519001_王澜洁_9.zip).

1. (Revise selection sort) The selection-sort function repeatedly finds the smallest number in the current list and swaps it with the first one. Rewrite this program by finding the largest number and swapping it with the last one. Write a test program that reads in ten numbers, invokes the function, and displays the sorted numbers. Here is the sample run:

```
Enter numbers: 5 7 3 9 18 23 1 36
[1, 3, 5, 7, 9, 18, 23, 36]
```

2. (Bubble sort) Write a sort function that uses the bubble-sort algorithm. The bubble-sort algorithm makes several passes through the list. On each pass, successive neighboring pairs are compared. If a pair is in decreasing order, its values are swapped; otherwise, the values remain unchanged. The technique is called a bubble sort or sinking sort because the smaller values gradually “bubble” their way to the top and the larger values “sink” to the bottom. Write a test program that reads in ten numbers, invokes the function, and displays the sorted numbers. Here is the sample run:

```
Enter numbers: 15 3 9 1 7 88 6
[1, 3, 6, 7, 9, 15, 88]
```

3. (Anagrams) Write a function that checks whether two words are anagrams. Two words are anagrams if they contain the same letters. For example, silent and listen are anagrams. The header of the function is:
def isAnagram(s1, s2):
(Hint: Obtain two lists for the two strings. Sort the lists and check if two lists are identical.) Write a test program that prompts the user to enter two strings and, if they are anagrams, displays is an anagram; otherwise, it displays is not an anagram.

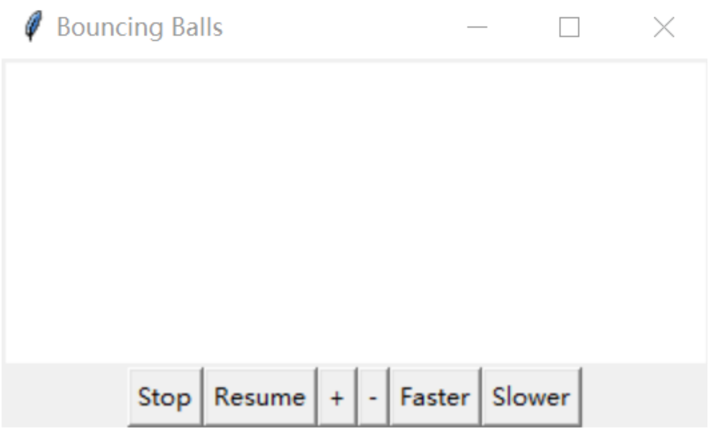
```
Enter the first string: silent
Enter the second string: listen
silent and listen are anagram.
```

4. (Occurrences of each digit in a string) Write a function that counts the occurrences of each digit in a string using the following header:
def count(s):
The function counts how many times a digit appears in the string. The return value is a list of ten elements, each of which holds the count for a digit. For

example, after executing counts = count("12203AB3"), counts[0] is 1, counts[1] is 1, counts[2] is 2, and counts[3] is 2.
Write a test program that prompts the user to enter a string and displays the number of occurrences of each digit in the string. Here is a sample run of the program:

```
Enter a string: 232534312
1 occurs 1 time
2 occurs 3 times
3 occurs 3 times
4 occurs 1 time
5 occurs 1 time
```

5. (Optional) (Tkinter: bouncing balls) Revise Bouncing Balls to add two buttons—Faster and Slower,—to speed up or slow down the ball movements.


Bouncing Balls
Stop
Resume
+
-
Faster
Slower