Task 4

Team Members: ____

In this practice, we will be implementing a the several sorting algorithms discussed in the class and experimenting with the runtime for these sorting algorithms. You are required to use python. For help you can use the lecture slides.

Task: (points 2.5+2.5+2.5+2.5)

1. Implement the quick sort algorithm in python to sort in ascending order.
2. Implement the radix sort algorithm in python to sort in ascending order.
3. Implement the heap sort algorithm in python to sort in ascending order.
4. Compare the time for these implementations for the following cases. The code for recording the runtime is provided below.

   Case A: Input to the sort algorithm is a sorted list (ascending order)
   Case B: Input to the sort algorithm is a sorted list (descending order)

   Also vary the size of the input list for your experimentation:
   1. 10,000
   2. 100,000
   3. 1000,000

   [Note: There are 6 possible cases to run for each algorithm.]

   For each run take three readings and populate the table in the next page.
5. Report your findings.

<u>Recording the runtime:</u>

To determine the sorting time, replace line 3 with appropriate function name.

```
1. import time # Doc available @ https://docs.python.org/3/library/time.html
2. t = time.process_time()
3. a = my_insertion_sort(n)
4. elapsed_time = time.process_time() – t
5. print(elapsed_time) # Note down this time in the table
```

Sorting Algorithm	Case	Reading	10,000	Avg	100,000	Avg	1,000,000	Avg
Quick Sort	A	1	0.046875		0.484375		2.703125	
WC: N^@	A	2	0.046875		0.453125	0.36979	2.953125	
AC: nlogn	A	3	0.03125	0.391	0.5		2.890625	2.895833
	B	1	0.046875		0.28125		3.015625	
	B	2	0.03125		0.234375		2.96875	
	B	3	0.03125		0.265625		2.84375	
Radix Sort	A	1	0.0		0.09375		1.265625	
	A	2	0.0		0.09375		1.28125	
	A	3	0.015625	0.007	0.078125		1.328125	1.4401
	B	1	0.0		0.109375	0.0963	1.5625	
	B	2	0.015625		0.09375		1.609375	
	B	3	0.015625		0.109375		1.59375	
Heap Sort	A	1	9.938642		1000.158		> hour	
	A	2	10.05995		1018.3865		> hour	
	A	3	9.969996	9.9194	1001.749	996.53	> hour	> hour
	B	1	9.938458		997.4946		> hour	
	B	2	9.802745		982.1778		> hour	
	B	3	9.8069939		979.267		> hour	

Note: It is expected that in some cases the sorting can take significantly long (in range of hours).