**Pranjit Kumar Kalita**
**Project 2**

| Shell Sort using Insertion Sort through linked lists | | | | | |
|---|---|---|---|---|---|
| File Name | # of Comparisons | # of Moves | I/O Time | Sorting Time | |
| 1000.txt | N/A | N/A | 0.000000e+00 | 1.000000e+00 | |
| 1000000.txt | N/A | N/A | 0.000000e+00 | 0.000000e+00 | |

Time Complexity for Shell_Sort function:
largest_index() function – O(n)
findSize() function – O(n)
power() function - O(n)
List_Insert() function - O(n^2)
Insertion_Sort() function – O(log n)
My program showed a very long Sorting time for 10000 and above elements despite implementing it with a linked list pointing to a linked list. Second, it showed me the same situation when I was initially implementing the function using repetitive iteration of the linked link of integers. My expectation was that using the list of list would change the time complexity drastically. But, since it didn't change anything much other than reducing the Sorting Time for 1000 elements from 2 seconds to 1 second, I am a little unsure as to what went wrong. Since I couldn't locate the problem, I submit to you my findings of the individual time complexities of the functions that my Shell_Sort() function implements.

**Space Complexity for Shell Sort using Insertion Sort**:
- allocate memory for linked list of linked list – O(n)
- variables – O(1)
- largest_index() function call – O(log n)

Inside the loops, no further memory is being allocated, while being deallocated before the next loop iterates. Only the linked list of nodes is being called each time to insert elements into it. Thus, space complexity is O(n).

Comparing my findings to the ones obtained from project 1, the I/O time and Sorting time for 10 elements are both zero. The Sorting time for 1000 elements is 1second for this implementation as opposed to 0s for project 1, maybe because of the iteration through the linked list that is required to get to the linked list element. Similarly, the extremely big times for 10000 and above elements could be explained through the same logic.