CS 2302  
Lab 2 – Option B  
Victor Huicochea

**Lab 2 – Option B**

**Introduction**

A text file containing 10 million pairs of usernames and passwords are given. A linked list must be created from those pairs, but every node of the linked list should have a “count” attribute, which will increase every time a password is repeated. For instance, if the same password is present 3 times, the linked list should have the password once with the count attribute set to 3.

The same list should be created using a dictionary. The key would be the password and the value related to that key would be the number of times the password is repeated.

There should also be a function for Bubble Sort and Merge Sort. These two would be used to sort the list from the most repeated password, to the least repeated one.

The program should return the 20 most repeated passwords.

**Proposed Solution**

The strategy used to solve the issue consisted in four steps:

1. Creating a function that reads the .txt file. It generates an array for every line that holds the username and password for that line. It then checks if the linked list already has that password so that it won’t hold any duplicates. If the password is already present, its “count” attribute will increase in 1 unit. If the password is not present, it will be added at the end of the linked list.
   1. This solution has a Big-O(n^2), where n is the number of passwords, or lines, present in the .txt file. It is O(n^2) because it has a loop that will run for every line (this is n times), and, each time a new password is read, the whole list will be traversed to make sure that the password is not present yet (this is n steps). Therefore, n \* n = n^2. It can be better understood by looking at the following table:

|  |  |  |
| --- | --- | --- |
| # **of Passwords (lines)** | **# of items in L. List** | **Total steps** |
| 100 | 100 | 100\*100 = 10, 000 |
| 1000 | 1000 | 1000\*1000 = 1,000,000 |
| 10 | 10 | 10\*10 = 100 |

1. Creating a function that reads the .txt file. It generates an array for every line that holds the username and password for that line. It then checks if the dictionary already has that password so that it won’t hold any duplicates. If the password is already present, the value linked to the password will increase by 1. Thus, each password is linked to the number of times it has been present in the file. If the password is not present in the dictionary, it will be added as a new key.
   1. This solution has a Big-O(n), where n is the number of passwords, or lines, present in the .txt file. It is O(n) because it has a loop that will run for every line, this is n steps. Inserting each password takes O(1) because each password is a unique key, which means that it is not necessary to look through the whole dictionary to find it. Therefore, n \* 1 = n. It can be better understood by looking at the following table:

|  |  |  |
| --- | --- | --- |
| # **of Passwords (lines)** | **# of steps needed to find PW** | **Total steps** |
| 1000 | 1 | 1000\*1 = 1,000 |
| 100 | 1 | 100\*1 = 100 |
| 10 | 1 | 10\*1 = 10 |

1. Creating a function following the Bubble Sort algorithm. The way it works, is that given a linked list, it takes the first two values (in this case is the number of times a password is present on the .txt file) and compares them. If the first value is smaller than the second one, the nodes will change place. This process is repeated until the smallest value is placed at the end of the list, and then the algorithm starts again from the top of the list.
   1. This solution has a Big-O(n^2), where n is the number of items in the linked list. It is O(n^2) because each item will be compared against every other item, this is n steps (number of items) time n steps (number of items each is compared to. Therefore, n \* n = n^2. It can be better understood by looking at the following table:

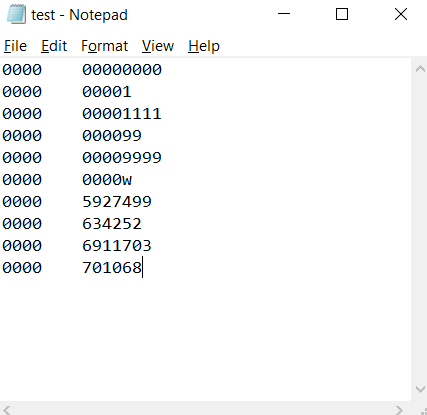
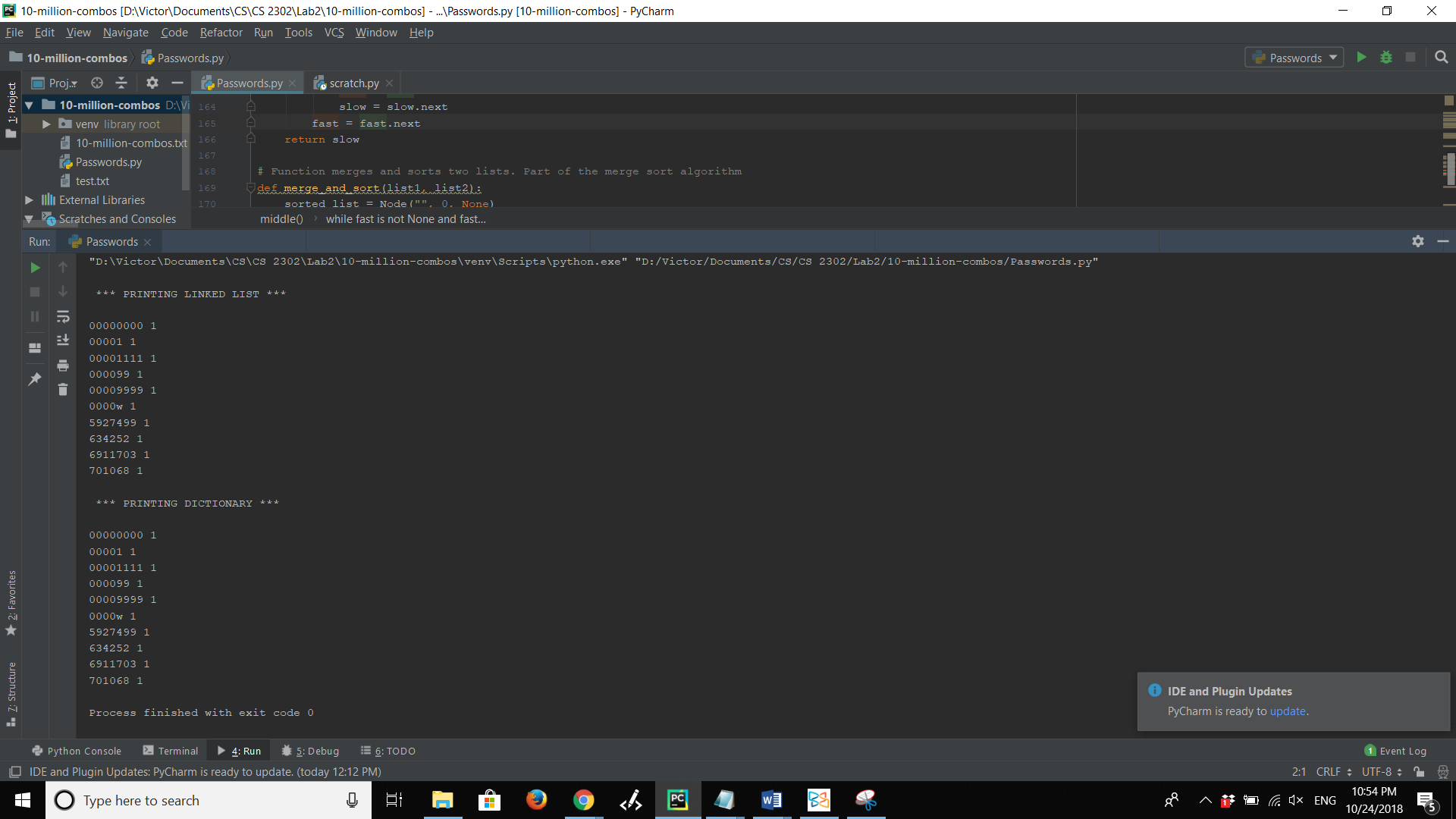
|  |  |  |
| --- | --- | --- |
| # **of items** | **# of items compared to** | **Total steps** |
| 1000 | 999 | 1000\*999 = 999,000 |
| 100 | 99 | 100\*99 = 9,900 |
| 10 | 9 | 10\*9 = 90 |

1. Creating a function following the Merge Sort algorithm. The way it works, is that given a linked list, it will be divided in half. This will be recursively repeated until these two lists consist of only one item each. Once each list consists of only one item, they will be merged back, comparing each value to decide which one would go first (sorting the list).
   1. This solution has a Big-O(nlogn), where n is the number of items in the linked list. It is O(nlogn) because first, finding the midpoint of each sub-list takes only O(1). Secondly, dividing the list in half takes logn number of steps. Finally, merging it takes n number of steps since it merges n number of elements. Therefore, 1 \* logn \* n = nlogn

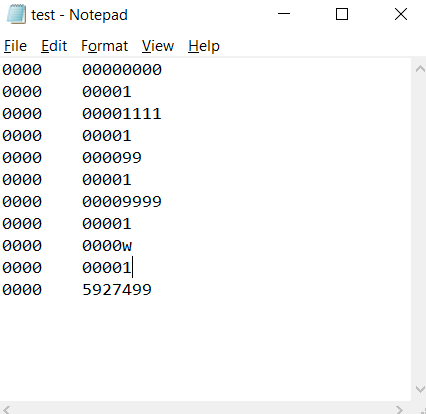
|  |  |  |  |
| --- | --- | --- | --- |
| **Steps finding mid** | **Steps dividing** | **Steps merging** | **Total steps** |
| 1 | 4 (list of 16) | 16 | 1\*4\*16 |
| 1 | 5 (list of 32) | 32 | 1\*5\*32 |
| 1 | 6 (list of 64) | 64 | 1\*6\*64 |

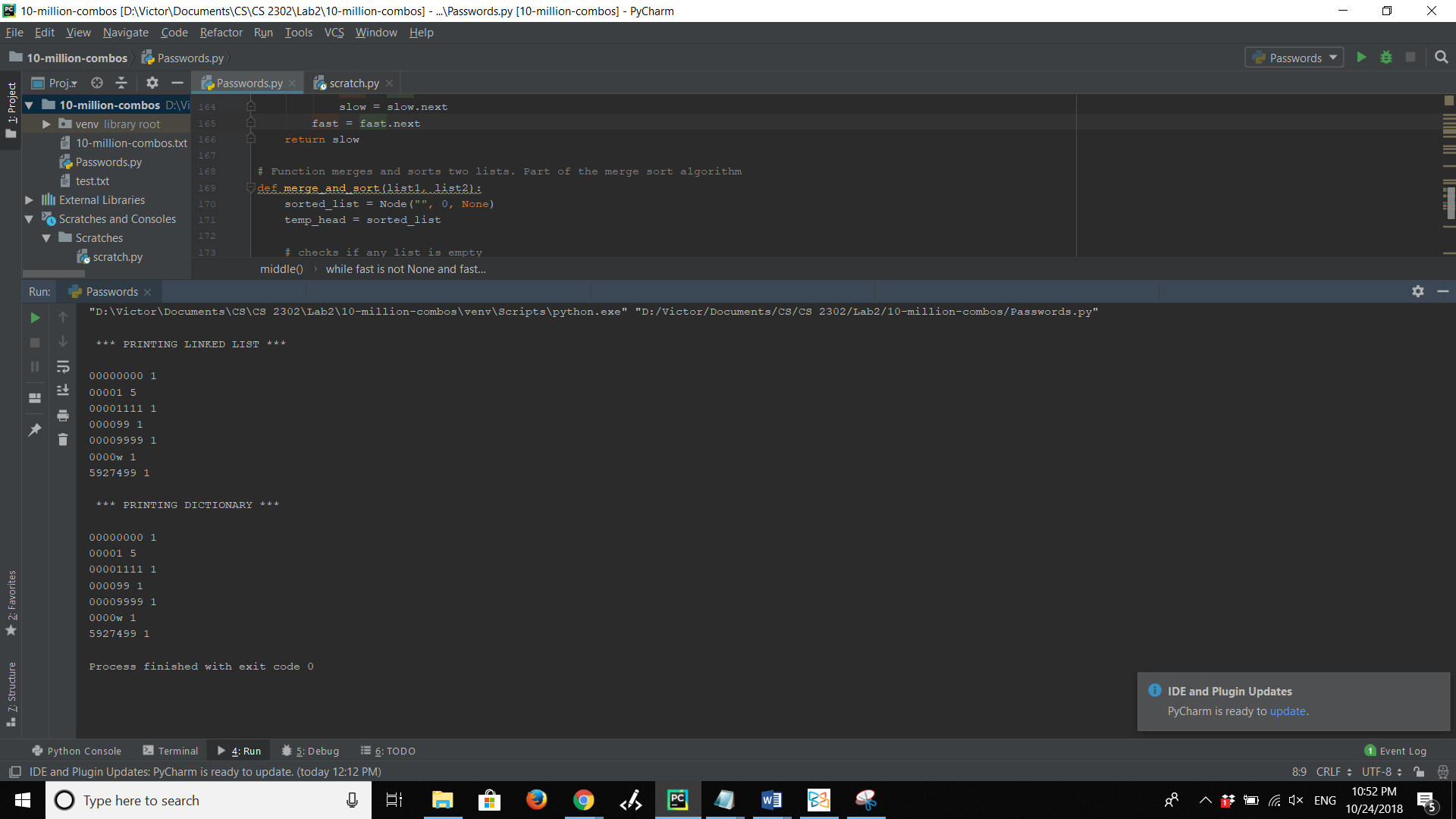
**Experimental Results**

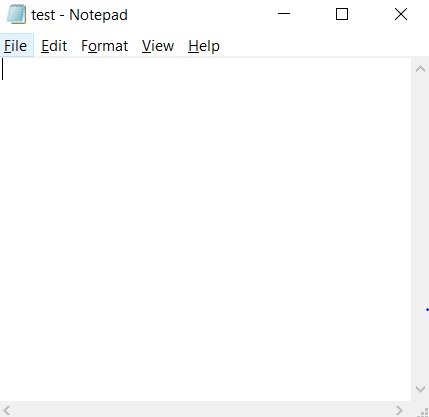
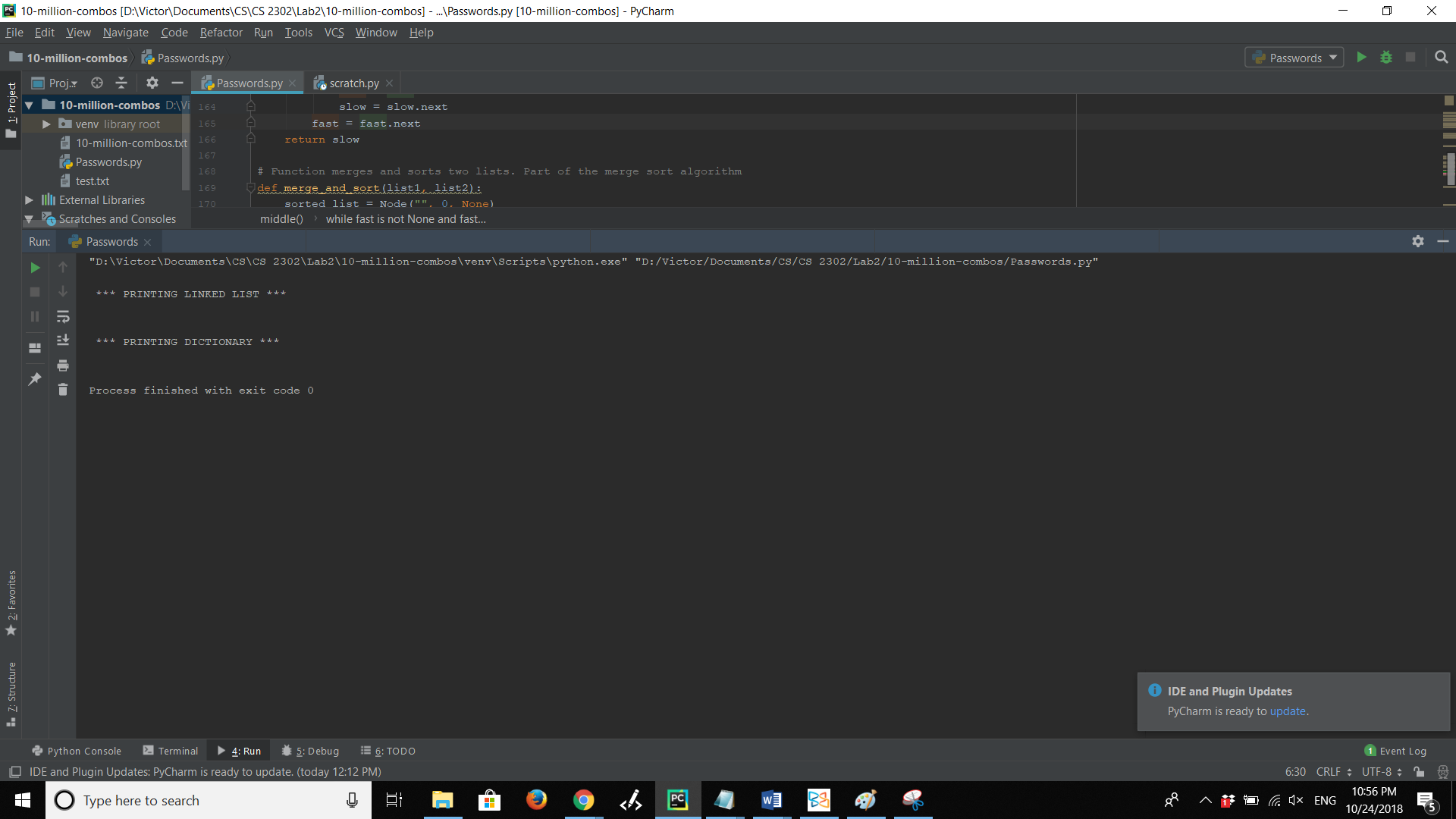
1. Testing with a portion of the lines provided by the instructor.
   1. The linked list and dictionary returned by solution a and solution b were printed, along with their “count” attribute, with the intention of showing that it correctly added new passwords.

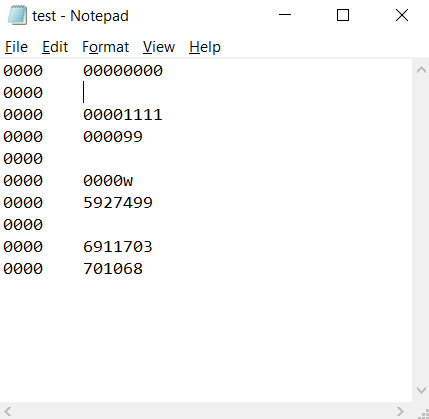
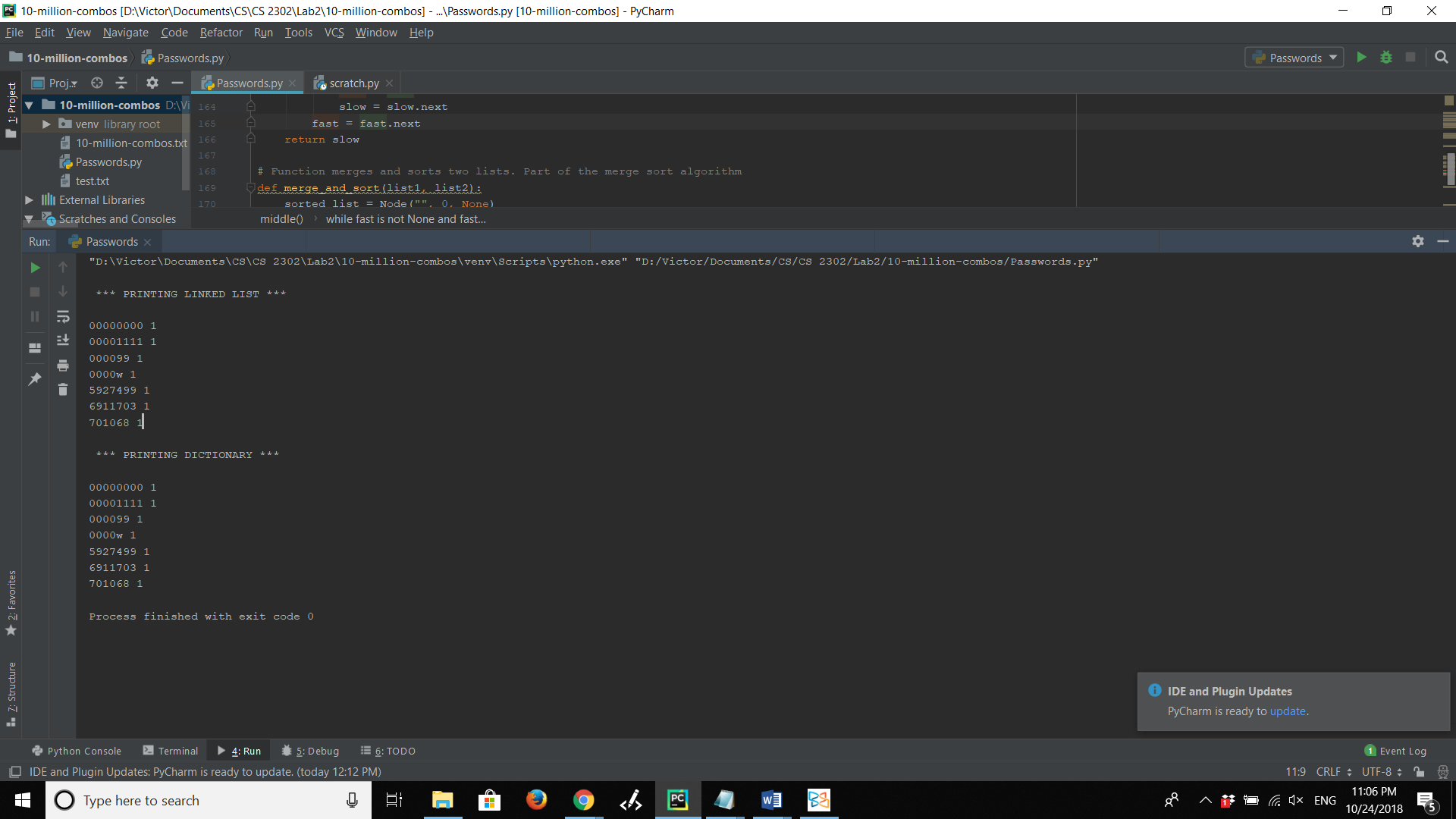


1. Testing with a repeated password.
   1. The linked list and dictionary returned by solution a and solution b were printed, along with their “count” attribute, with the intention of showing that it correctly added new passwords and counted when a password is present more than once.





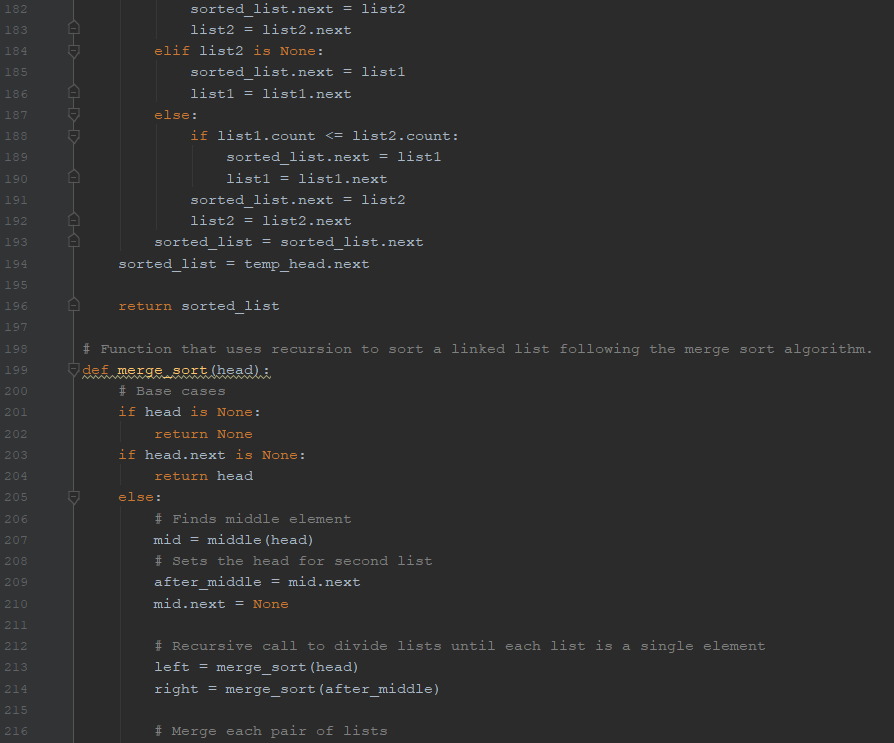
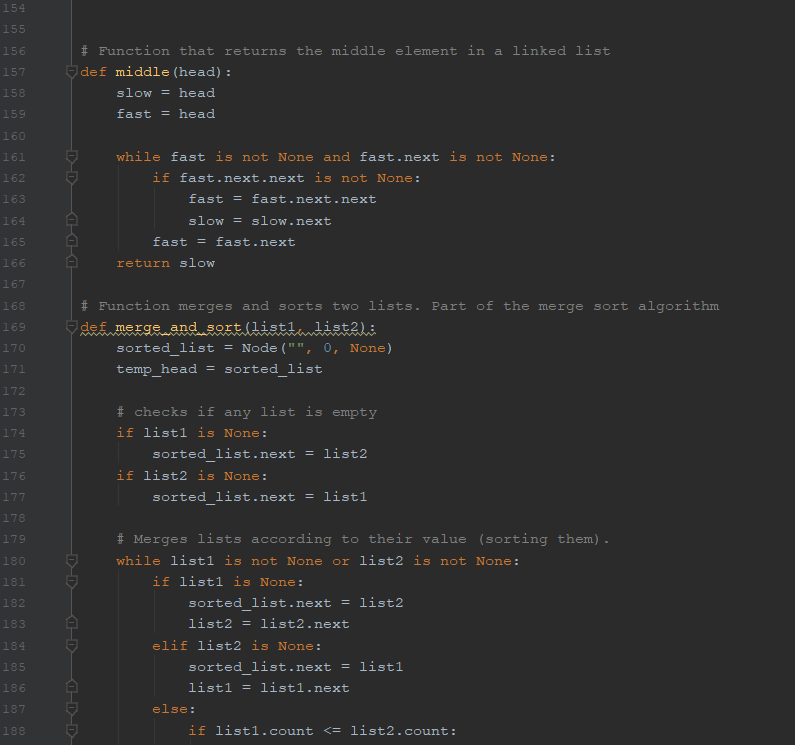
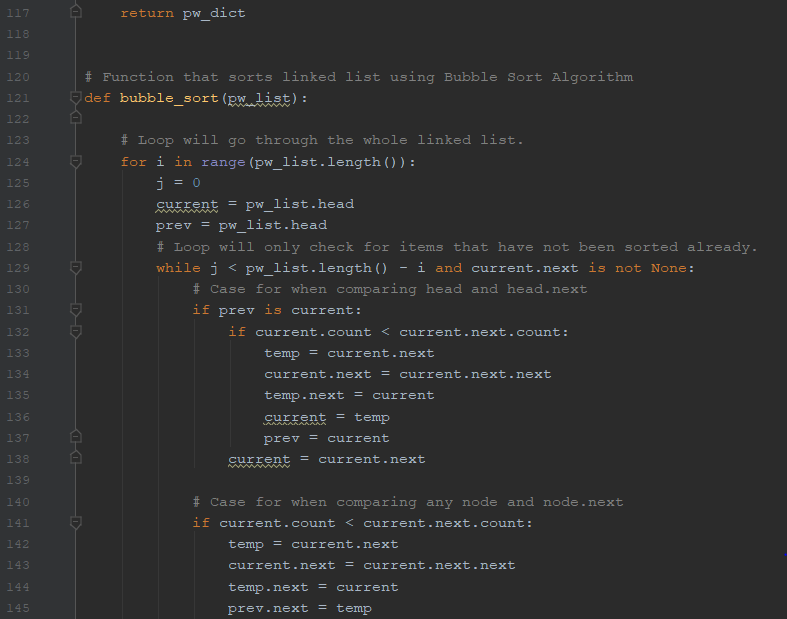
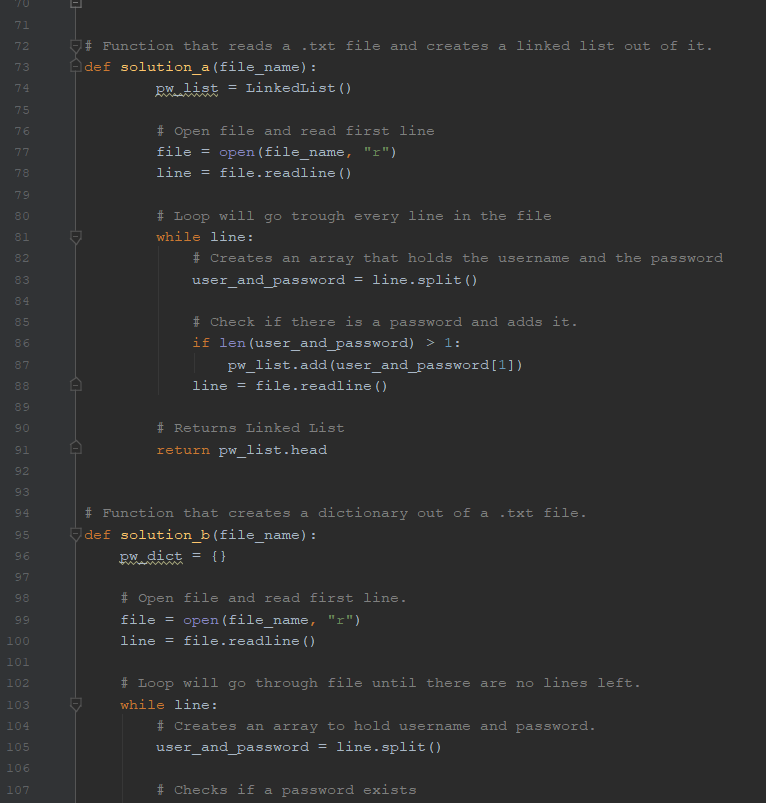
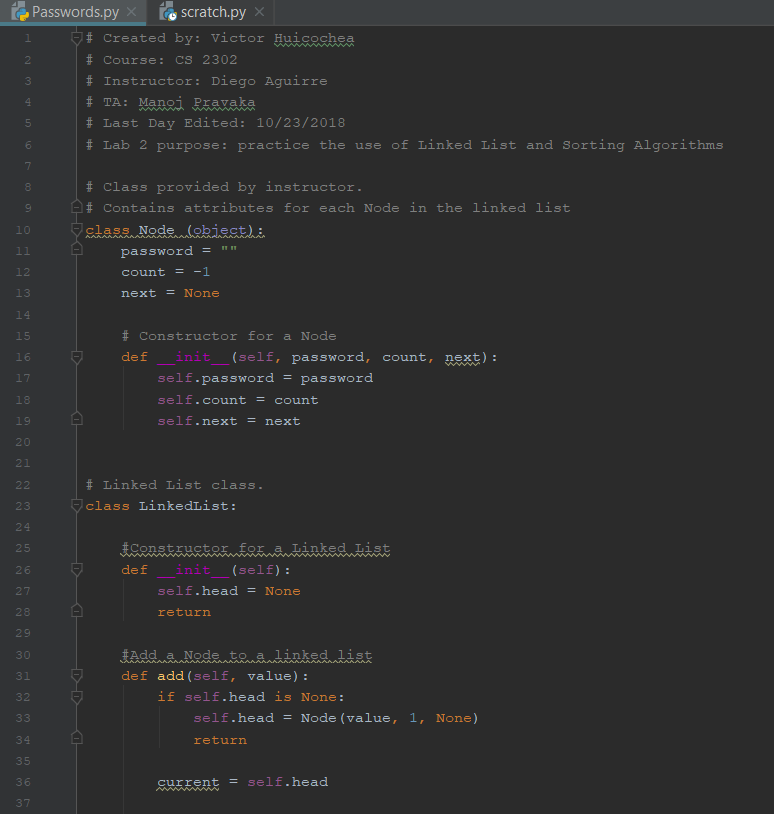
1. Testing with an empty file.
   1. When attempting to print the linked list and dictionary of an empty file, nothing gets printed on the console. The purpose of this test was to make sure that no error would be found with empty files.
2. Testing with lines that only have usernames.
   1. This test demonstrate that no error is shown when any line in the .txt file does not contain a password.

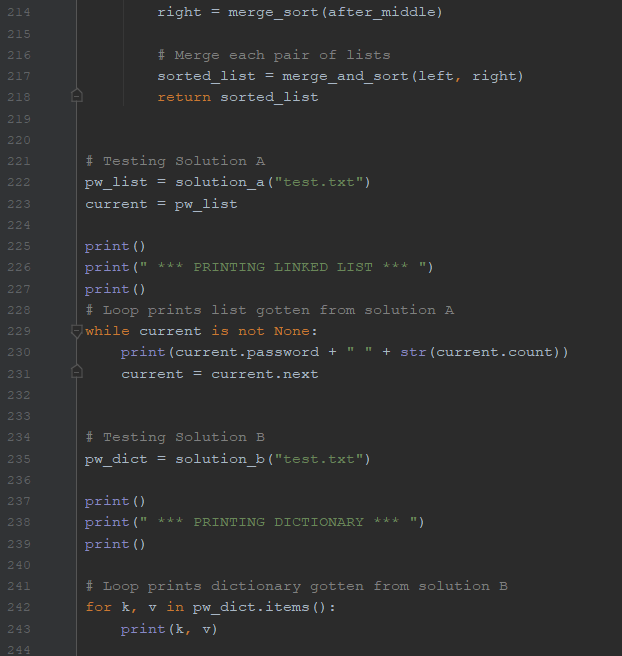


**Conclusions**

This project helped me to fully understand how linked lists work and how useful they can be. It is now easier for me to visualize how information or nodes move when changing pointers.

I also understood how bubble and merge sort work, and now I know where their Big-O notations come from. However, I had a really hard time when coding these algorithms, in fact, I could never get them to work. Probably I could later meet with the TA or the professor to ask for advice, because it is hard for me that even though I understand the logic perfectly well, I cannot make those two functions work.

**Appendix – Source Code**

****

**Academic Honesty Certification**

****I certify that this project is entirely my own work. I wrote, debugged, and tested the code being presented, performed the experiments, and wrote the report. I also certify that I did not share my code or report or provided inappropriate assistance to any student in the class.