Week 5 Homework
Matt Strayer

Consider the following list of integers: [1,2,3,4,5,6,7,8,9,10]. Show how this list is sorted by the following algorithms:

- **bubble sort**
- **selection sort**
- **insertion sort**

For the bubble sort, the program will run through the list starting at the beginning and repeatedly compare neighboring elements in the list and will swap if the element that comes first in the pair is greater than the second element. So it will check 1 and 2 in our list, since 1 isn't larger than 2, no swap is done and they stay in the same place. Next it moves and checks 2 and 3, and once again no swap is initiated because 2 is less than 3. It will continue to do this and make passes until the list is sorted. This method effectively 'bubbles' up the largest element to the end of the list. The first pass of the list the algorithm makes moves the largest element to the end of the list and the second pass bubbles up the second largest element and so on.

Index Location

0	1	2	3	4	5	6	7	8	9
1	2	3	4	5	6	7	8	9	10

Element

For the selection sort, the program will repeatedly go through the list and pick the largest element and put it into the right position. It improves upon the bubble sort because it only makes one swap each pass and is more efficient that way. For our provided list, it will pass through analyzing each element, starting with 1 and working its way up to 10. Since the list is in order it will get to the 10 last, identify it as the maximum value and set it in index location 9. It would then make a new pass and identify 9 as the next largest and put it in its rightful place, this time being index location 8.

The insertion sort works a little differently and creates a sub-array that it adds to as it makes passes through the given list. It will start with the beginning of the list and create its sub list there. It starts with the first element and then makes its way down the list and adds a new element to the sub list as it gets to it. The element is then inserted into its proper place in the sub list based on size. So for our list it would start by adding 1 to its sublist at location 0. It will then go to 2 and add it to the sublist and make sure it's in the right location. Since 2 is larger than 1 it stays in index location 1. If the next element had been smaller than the first element in the sub list, it would have been inserted into index location 0.

See program files for more information.

Research perfect hash functions. Using a list of names (classmates, family, etc. generate the hash values using the perfect hash algorithm.

I did some research on perfect hash functions and saw a lot of different algorithms that people use to generate hash values. One that stuck out to me was the ‘hashlib’ module that comes with Python 3. This takes the names from the list of family names I created:

```
 familyNamesList = [ "Matt", "Megan", "Colby", "Mary", "Todd", "Jane", "Kenzie", "Evan" ]
```

It then uses the SHA - 256 algorithm and for each name in the list it provides an alphanumeric hash value for that element in the list. I liked this one because it uses cryptography and is used in a lot of different technologies.

It produces these hash values:

```
Matt  - Hash Value:
84a4b19e19aa4e2a562ae0286b1e188ef4f4f9a98a92b8730d20a1e0f2882523
Megan  - Hash Value:
2fcc60c2ab3be068dc8bfa79d2431e62343435aaabb3ae8c45ae845306623b84
Colby  - Hash Value:
404cb1e9ca7b5c693cf845ba207d4c55f9596a1e98fc2d9563664736711a7818
Mary  - Hash Value:
aebac53c46bbeff10fdd26ca0e2196a9bfc1d19bf88eb1efd65a36151c581051
Todd  - Hash Value:
f79b56140024d7668fa8d7df162c4de35bfda3d225a115e29589195d071081ec
Jane  - Hash Value:
4f23798d92708359b734a18172c9c864f1d48044a754115a0d4b843bca3a5332
Kenzie  - Hash Value:
03b1963f7e55c30619c02e168ce8837a4e2c8fe58a22d3b3897e4a94f2c3e5b4
Evan  - Hash Value:
832c5551378ad56e0181af310d69ab9fb2f776ee6fb384b3e6673d08fc93526e
```

I also did a simple function using the built-in ‘hash()’ function in python and used that to assign hash values to the name in my list. It produced the values below:

```
Matt -4454684382022666693
Megan 906825526039876139
Colby -971502285988417834
Mary -4512184809422975525
```

```
Todd -5562355654545705958
Jane -1226477695741598771
Kenzie -2935614149041543106
Evan -3328346063627769778
>>>
```

Please see both of the hash program files for more info.