PLAYBOOK NOTES: INSERTION SORT

You Tube Video: https://www.youtube.com/watch?v=8mJ-OhcfpYg

Or you can search you tube for *Bro Code Insertion Sort*

The algorithm is patterned after Program 13.21 in the textbook and is consistent with the above YouTube video.

("Data Structures, Algorithms & Software Principles in C", Thomas A. Standish, © 1995)

Algorithm:

```
Integer tempValue and tempIndex

For loop: index → 1…length-1
{
  Set tempValue to the value at array position index
  Set tempIndex to equal index - 1

  test1 = Check to see if tempIndex is greater than or equal to 0
  test2 = Check to see if array at position tempIndex is
          greater than the value stored in tempValue

   Loop while test1 equals true and test2 equals true
   {
      Assign array at position tempIndex + 1 to array at position tempIndex
      Decrease tempIndex by 1
   }

   Set array at position tempIndex + 1 to equal tempValue
}
```

EXAMPLE:

Before we begin, notice the outside for-loop starts at 1 and not 0. This is NOT a mistake.

Given the array of 4 integers below:

Index 0	Index 1	Index 2	Index 3
80	16	8	13

FIRST TIME THROUGH THE FOR-LOOP:

Start by putting the value at index 1 in a temp storage:

Index 0	Index 1	Index 2	Index 3
80		8	13

Temp:	16

Looking at the values to the left (second loop), we compare each value (in this case there is only the 80) to our temp value.

If the value is greater than our temp value, move it to the right one spot.

Do this until a value is NOT greater than our temp value OR we run out of elements to compare.

80 is greater than 16, so we move the 80 to the right.

Now we are out of elements to look at.

Index 0	Index 1	Index 2	Index 3
	80	8	13

Temp:	16

The second loop is done, so we put the 16 in the empty spot.

Index 0	Index 1	Index 2	Index 3
16	80	8	13

Temp:	

SECOND TIME THROUGH THE FOR-LOOP:

Index 0	Index 1	Index 2	Index 3
16	80	8	13

Temp:	

Start by putting the value at index 2 in a temp storage:

Index 0	Index 1	Index 2	Index 3
16	80		13

Temp:	8

Looking at the values to the left (second loop), we compare each value to our temp value.

If the value is greater than our temp value, move it to the right one spot.

Do this until a value is NOT greater than our temp value OR we run out of elements to compare.

80 is greater than 8, so we move it one spot to the right.

16 is greater than 8, so we move it one spot to the right.

Now we are out of elements to look at.

Index 0	Index 1	Index 2	Index 3
	16	80	13

Temp:	8

The second loop is done, so we put the 16 in the empty spot.

Index 0	Index 1	Index 2	Index 3
8	16	80	13

Temp:	

THIRD TIME THROUGH THE FOR-LOOP:

Index 0	**Index 1**	**Index 2**	**Index 3**
8	16	80	13

Temp:	

Start by putting the value at index 3 in a temp storage:

Index 0	**Index 1**	**Index 2**	**Index 3**
8	16	80	

Temp:	13

Looking at the values to the left (second loop), we compare each value to our temp value.

If the value is greater than our temp value, move it to the right one spot.

Do this until a value is NOT greater than our temp value OR we run out of elements to compare.

80 is greater than 13, so we move it one spot to the right.

16 is greater than 13, so we move it one spot to the right.

16 is NOT greater than 13, so we are done with the second loop.

Index 0	**Index 1**	**Index 2**	**Index 3**
8		16	80

Temp:	13

The second loop is done, so we put the 13 in the empty spot.

Index 0	**Index 1**	**Index 2**	**Index 3**
8	13	16	80

Temp:	

We have now completed the outside loop and our array is sorted!

Analysis:

The Insertion sort runs in time O(n^2)