SCREENSHOT OF OUTPUT:

```
elapsed time for bubble sort 34.656
elapsed time for radix sort 0.018
elapsed time for cocktail sort 27.816
BubbleSort, RadixSort, and CocktailSort all produce the same sorted arrays.
```

Describe how the runtimes correspond to the Big O numbers for each. Do the number correlate or not?

Assumptions:

For BubbleSort, I assumed a sort into ascending order, that it was a fixed, comparable array of a size of 100,000, and that after every loop the value at the last index was properly sorted. In RadixSort, I assumed a sort into ascending order, that only integer values could be used, and that the max number had the most significant digits; therefore, the max number held the number of times the rSort would be called. Since I assumed RadixSort could only use integer values, it had a fixed, integer array of size 100,000. In CocktailSort, I made all the same assumptions as BubbleSort, but I also assumed that after the array was sorted from the right to left that the minimum was in the smallest accounted for index.

Sort Type	Big O	Runtime (N = 100,000)	Runtime (N = 200,000)	Runtime (N = 400,000)
Bubble	$O(N^2)$	34.656	138.624	554.496
Radix	*O((N+10)*log(k))	0.018	0.03599	0.07199
Cocktail	$O(N^2)$	27.816	111.264	445.056

* = where k is the number of significant digits in the max value

Running time = a * O(N)

BubbleSort:
$34.656 = a * O(100{,}000^2)$
$a = 3.4656 \times 10^{-9}$

RadixSort:
$0.018 = a * O((100{,}010) * \log(6))$
$a = 2.3129 \times 10^{-7}$

CocktailSort:
$27.816 = a * O(100{,}000^2)$
$a = 2.7816 \times 10^{-9}$

As calculated in the work above, the running times correspond to the Big O of 3.4656×10^{-9} for BubbleSort, 2.3129×10^{-7} for RadixSort, and 2.7816×10^{-9} for CocktailSort. Which correlate, because given different values for N, when we estimate for each sort as N grows larger, BubbleSort is the slowest, CocktailSort is second, and RadixSort is the fastest.