Step 1: Runtime Analysis (Extra Credit included)

- extraLargeArray:
  Insert= 843ms
  Append= 3ms
- largeArray:
  Insert= 6ms
  Append= 553μs
- mediumArray:
  Insert= 398μs
  Append= 198μs
- smallArray:
  Insert= 124μs
  Append= 116μs
- tinyArray:
  Insert= 56μs
  Append= 105μs

In the extraLarge array there are 1,000,000 items put into an array starting at 0. When passed into the doubleInsert function, it takes 843ms because the array is not only adding a number but also unshifting the whole array in order to add the new number to become the 1st index. When passing the extraLargeArray into the doubleAppend function it takes 3ms due to the new numbers being added to the end of the array only. Here we see that due to the array using the unshift method in the doubleInsert function it takes a longer time when compared to the push method in the doubleAppend function. This pattern continues for the LargeArray, mediumArray and smallArray, but as the arrays decrease so does the time it takes for the doubleInsert and doubleAppend functions to run. When looking at the different times you can also see that as the array gets smaller the times for the insert and append come closer together, for example: in the extraLargeArray insert is 843ms while append is 3ms and in the smallArray insert is 124μs and append is 116μs. Once we get to the tinyArray insert becomes faster than append due to the small length of the array compared to the much larger ones. The larger the array the longer it will take for push but for unshift that time will be even longer due to the array needing to be unshifted to the left. For this reason the doubleAppend function scales better than the doubleInsert function.