```
#1: When using the last element of the array for the partition value the worst
possible case for quick sort arises when the partition makes no swaps
#aka the array is sorted in ascending or descending order
#when this is the case, quick sort requires n-1 total runs with n-1-runs
comparisons, which ends up being the same as bubble sorts average case with n(n-
1)/2

#2: a vector with 16 elements that could represent this case simply counts from 0
to 15, [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15] though for demonstration purposes
it is easier
#to choose a reverse sorted list [15,14,13,12,11,10,9,8,7,6,5,4,3,2,1,0]
#after first partition the list will be [0,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1]
#followed by recalls with sub sections of the list ass follows [0]
[14,13,12,11,10,9,8,7,6,5,4,3,2,1]
#following this trend all the way through we get
#[1,15,14,13,12,11,10,9,8,7,6,5,4,3,2]
#[1] [15,14,13,12,11,10,9,8,7,6,5,4,3,2]
#[2,15,14,13,12,11,10,9,8,7,6,5,4,3]
#[2] [15,14,13,12,11,10,9,8,7,6,5,4,3]
#[3,15,14,13,12,11,10,9,8,7,6,5,4]
#[3][15,14,13,12,11,10,9,8,7,6,5,4]
#[4,15,14,13,12,11,10,9,8,7,6,5]
#[4][15,14,13,12,11,10,9,8,7,6,5]
#[5,15,14,13,12,11,10,9,8,7,6]
#[5][15,14,13,12,11,10,9,8,7,6]
#[6,15,14,13,12,11,10,9,8,7]
#[6][15,14,13,12,11,10,9,8,7]
#[7,15,14,13,12,11,10,9,8]
#[7][15,14,13,12,11,10,9,8]
#[8,15,14,13,12,11,10,9]
#[8][15,14,13,12,11,10,9]
#[9,15,14,13,12,11,10]
#[9][15,14,13,12,11,10]
#[10,15,14,13,12,11]
#[10][15,14,13,12,11]
#[11,15,14,13,12]
#[11][15,14,13,12]
#[12,15,14,13]
#[12][15,14,13]
#[13,15,14]
#[13][15,14]
#[14,15]
#[14][15]
#resulting in the final list[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
```

```
#4: The results match our complexity analysis with a quadratic curve modling the
results of comparisons against number of elements in the worst case, similar to
that of the average case of bubble sort
```