1.

Method	Vector.cs	Mecrosoft .Net framework
Count()	O(1)	O(1)
Capacity()	O(1)	O(1)
Add()	O(1) if count< capacity O(n) if count>= capacity	O(1) if count< capacity O(n) if count>= capacity
IndexOf()	O(n)	O(n)
Insert()	O(n)	O(n)
Clear()	O(n)	O(n)
Contains()	O(n)	O(n)
Remove()	O(n)	O(n)
RemoveAt()	O(n)	O(n)

2

The below function is O(n log(n)) and Ω(n log(n)), therefore, it's θ(n log n)

```
private void MergeSort<K>(K[] sequence, IComparer<K> comparer) where K :
IComparable<K>
      {
            int n = sequence.Length;
            //when array length is 1, it is sorted
            if (n < 2) return;

            //chop chop
            int mid = n / 2;
            K[] s1 = sequence.Take(mid).ToArray();
            K[] s2 = sequence.Skip(mid).ToArray();

            //conquer, sort the 2 sequences recursively
            MergeSort(s1, comparer);
            MergeSort(s2, comparer);

            //Merge result to original
            Merge(s1, s2, sequence, comparer);
       }
```

The name of this function is Merge Sort.

3

a. f = x^1/2
   g= log(n)

$$\lim_{n\to\infty} \frac{n^{\frac{1}{2}}}{\log n} = \infty$$

   f grows faster than g
   f ∈ Ω(g)

b. f = 1500
   g = 2

$$\lim_{n\to\infty} \frac{1500}{2} = 750$$

   f is 750 times bigger than g, they grow at the same rate (f = 750g)
   f ∈ θ(g)

c. f = 800*2^n
   g = 3^n

$$\lim_{n\to\infty} \frac{800 * 2^n}{3^n} = 0$$

   f grows slower than g
   f ∈ O(g)

d. f = 4^(n+13)
   g = 2^(2n+2)

$$\lim_{n\to\infty} \frac{4^{n+13}}{2^{2n+2}} = 16777216$$

   f is 16777216 bigger than g, they grows at the same rate (f(n) = 16777216 * g(n))
   f ∈ θ(g)

e. f = 9n log(n)
   g = n log(9n)

$$\lim_{n\to\infty} \frac{9n * log(n)}{n * \log(9n)} = 9$$

   f is 9 time bigger than g, they grows at the same rate (f(n) = 9 * g(n))
   f ∈ θ(g)

f. f = O(g)
   f = n!
   g = (n+1)!

$$\lim_{n\to\infty} \frac{n!}{(n+1)!} = 0$$

   f grows slower than g
   f ∈ O(g)