Runtime and Big O

1. Put the following runtimes in increasing order - O(n), O(log(n)), O(n!), O(n^2), O(nlog(n)), O(2^n)

2. What are the big-O complexities of the following functions?

```
a. def howManyNumbers(s):
       numbers = "1234567890"
       count = 0
       for char in s:
           for num in numbers:
               if (char == num):
                   count = count + 1
       return count
```

```
b. def containsVowel(s):
       for c in s:
               If c in [“a”, “e”, “i”, “o”, “u”]:
                   return true
       return false
```

```
c. def f(L): # L is a list with length n
          lenList = len(L)
          count = 0
          for i in range(lenList):
                   for j in range(lenList):
                        count += L[i]
          return count
```

```
d. def g(s): #s is a string of length n
            result = 0
            for char in string.ascii_lowercase:
                   if char in s:
                       s = s[1:]
                      result += 1
           return result
```

```
e. def h(L): #L is a list with length n
              i = 1
```

```
 listLength = len(L)
 result = []
 while i < listLength:
     result += L[i]
     i *= 3
return i
```

3. Given the following function containsVowels that checks if a string contains any vowels, what are the best and worse case runtimes? What are the runtimes of each?

```
def containsVowel(s):
   for c in s:
         If c in [“a”, “e”, “i”, “o”, “u”]:
            return true
   return false
```