1. (10 Points) Fill in the blanks by selecting the statements that can be true based on the statement in the first column.

	g(n) grows slower than f(n)	g(n) grows the same rate as f(n)	g(n) grows faster than f(n)
$f(n)=O(g(n))$	F	T	T
$f(n)=o(g(n))$	F	F	T
$f(n)=\Omega(g(n))$	T	T	F
$f(n)=\omega(g(n))$	T	F	F
$f(n)=\theta(g(n))$	F	T	F

2. (10 Points) Group the following functions f1, f2, …, f10 into different groups, so that functions within the same group grow at the same asymptotic rate. Also list groups in increasing asymptotic growth rate order. Note logn has base 10 and lgn has base 2.

f1	1 + 2 + 3 + … + (n-1) + n	n*(n+1)/2
f2	nlogn	nlogn
f3	(lgn)*(lgn)	(lgn)*(lgn)
f4	1 + 2 + 4 + 8 + … + 2^m ($n=2^m$)	2n-1
f5	2lgn + 100	2lgn + 100
f6	64n + 32	64n + 32
f7	$2^{(2n)}$	$2^{(2n)}$
f8	10logn + 3	10logn + 3
f9	2^n	2^n
f10	log(n!)	nlogn

Answer:
f5, f8
f3
f4, f6
f2, f10
f1
f9
f7

3. (20 Points) Provide best-case and worst-case running time and space complexity analysis in Big-Oh notation for **sort** method.

	Big-O Notation	Brief Explanation
Best-Case Running Time	$O(n)$	When the input array is already sorted, after the first for loop isOrdered is true, and the method returns. The first for loop is linear in running time.
Worst-Case Running Time	$O(n^2)$	When the input array is not in sorted order, the execution will go to the second for loop, which is $O(n^2)$ in running time.
Best-Case Space Complexity	$O(1)$	The method creates a constant number of variables without any method calls.
Worst-Case Space Complexity	$O(1)$	The method creates a constant number of variables without any method calls.

```java
// assume input array has at least 2 elements
void sort(int[] arr) {
      int n = arr.length;
      boolean isOrdered = true;
      for (int i=0; i<n-1; i++) {
            if (arr[i] > arr[i+1]) {
                  isOrdered = false;
                  break;
            }
      }

      if (isOrdered) return;

      for (int i = 0; i < n - 1; i++) {
            for (int j = 0; j < n - i - 1; j++) {
                  if (arr[j] > arr[j + 1]) {
                        int temp = arr[j];
                        arr[j] = arr[j + 1];
                        arr[j + 1] = temp;
                  }
            }
      }
}
```

4. (20 Points) Provide best-case and worst-case running time and space complexity analysis in Big-Oh notation for **the second method**.

	Big-O Notation	Brief Explanation
Best-Case Running Time	O(1)	The running time for the first two statements in the second method is O(1), and we need to analyze the running time of the third statement, calling the first method. When m is set to 50 from random number generation in the first method, the running time for the first method is O(1), so the running time for the second method is O(1) as well.
Worst-Case Running Time	O(n)	If m is never 50 when the first method is called, the first method will recursively call itself n times, thus running time is O(n). That gives us the running time for the second method is O(n).
Best-Case Space Complexity	O(1)	The space complexity for the first two statements in the second method is O(1), and we need to analyze the space complexity of the third statement, calling the first method. When the second method called the first method, if m is set to 50 from random number generation, the space complexity for the first method is O(1), so the space complexity for the second method is O(1) as well.
Worst-Case Space Complexity	O(n)	When the first method is called, if m is never 50, space complexity for the first method is O(n) due to n recursive calls results n stake frames on the system stack. Thus the space complexity for the second method is O(n).

```
public static void genCode(int n, List<Integer> list, Random random) {
        int m = random.nextInt(100);
        if (m != 50 ) {
         if (n == 0) {
              return;
         } else {
              list.add(m);
               genCode(n-1, list, random);
         }
        }
}

public static void genCode(int n) {
     List<Integer> list = new LinkedList<Integer>();
     Random random = new Random();
     genCode(n, list, random);
}
```

Submission Note

1) For written part of the questions:
    a) Write your answers inside a text document (in plain text, MS Word, or PDF format)
    b) Name the file as firstname.lastname.assignment1.txt(doc, docx, or pdf) with proper file extension
2) Due Sep 22, 11:59 PM