ICS 353 HW 1 Solution

1. Question 1.1 (a).

   6 comparisons.

2. Question 1.12.

Iteration	**Array**	**Number of comparisons**
1	[11,12], [1,5], [3,15], [4, 10],[2,7],[9,16],[8,14],[6,13]	8
2	[1,5,11,12], [3,4,10, 15], [2,7,9,16], [6, 8, 13, 14]	9
3	[1,3,4,5,10,11,12,15], [2,6,7,8,9,13,14,16]	14
4	[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16]	15
Total		48

3. Question 1.13.

   We know that the insertion sort algorithm performs $\frac{n \log n}{2}$ comparisons in the best case and $n \log n - n + 1$ comparisons in the worst case. If $n = 8$, then the minimum number of comparisons is 12 and the maximum number of comparisons is 17.

4. Question 1.13.

$f(n)$	$g(n)$	$f(n) = O(g(n))$	$f(n) = \Omega(g(n))$	$f(n) = \Theta(g(n))$
$2n^3 + 3n$	$100n_2^2 n + 100$	F	T	F
$50n + \log n$	$10n + \log\log n$	T	T	T
$50n \log n$	$10n \log\log n$	F	T	F
$\log n$	$\log^2 n$	T	F	F
$n!$	5^n	F	T	F

5. Question 1.15

   a) $2n + 3\log^{100} n = \Theta(n)$.

   To show that n dominates $\log^{100} n$ consider the following limit. $\lim_{n\to\infty} \frac{n}{3\log^{100} n} = \frac{\infty}{\infty}$.

   By applying L'Hopital's rule 100 times we get, $\lim_{n\to\infty} \frac{2(\ln 2)^{100} n}{3(100!)} = \infty$.

   b) $7n^3 + 1000n \log n + 3n = \Theta(n^3)$ since clearly n^3 dominates $n \log n$ and n.

   c) $3n^{1.5} + (\sqrt{n})^3 \log n = \Theta((\sqrt{n})^3 \log n)$. Note that $(\sqrt{n})^3 = n^{1.5}$.

6. Question 1.22

   We need to show that $\lim_{n\to\infty} \frac{2^n}{n^{100}} = \infty$.

   By applying L'Hopital's rule 100 times we get

   $\lim_{n\to\infty} \frac{(\ln 2)^{100} 2^n}{100!} = \infty$. Thus $n^{100} = O(2^n)$ but $2^n \neq O(n^{100})$,

7. Question 1.29

   $(\frac{1}{2})^n \prec 5 \prec \log\log n \prec \log n^{100} \prec \log^2 n \prec n^{1/100} \prec \sqrt{n} \prec n \log n \prec (\sqrt{n})^n \prec n! \prec 2^{n^2}$

8. Question 1.33

   b) In the worst case, the inner loop will be fully iterated when n is a power of two. In this case the algorithm time complexity can be computed as follows. Consider the inner loop, let $2^k = \frac{n}{2}$ and $r = \log j = k, k-1, ..., 1, 0$. Then, $\sum_{i=1}^{n} \sum_{r=0}^{k} 1 = n(k+1) = n(\log n - 1) = O(n \log n)$.

   c) In the worst case, when $\lfloor \frac{n}{2} \rfloor$ is an odd number the algorithm will run the inner loop only one time. Thus, $\sum_{i=1}^{n} 1 = n = \Omega(n)$.

   d) In the worst case, this algorithm is $O(n \log n)$ and $\Omega(n)$, which implies that, in the worst case, it is not $\Theta(n \log n)$ and $O(n \log n)$ is more appropriate the express the time complexity.