CS 141 Assignment #1

Ruiwen He SID:861203571

1. The following pseudo code shows an implementation of the selection sort algorithm.

```
1: function SELECTION-SORT(A, n)
2:     for i = 1 to n-1 do                C1
3:         min ← i                        C2
4:         for j = i + 1 to n do          C3
5:             if A[j] < A[min] then      C4
6:                 min ← j                C5
7:             end if
8:         end for
9:         swap A[i], A[min]              C6
10:    end for
11: end function
```

(a) Compute the *worst* case running time using the method shown in class for insertion sort. That is, assign a different constant to each of the lines 2-10 and use them to compute the running time.

Step	Cost
C1	n
C2	n-1
C3	$\sum_{j=2}^{n} T_j$
C4	$\sum_{j=2}^{n} T_j - 1$
C5	$\sum_{j=2}^{n} T_j\prime$
C6	n-1

Worst case: when number is ordered from large to small

$C1 \times n + (C2 + C6) \times (n-1) + C3 \times \sum_{j=2}^{n} T_j + C4 \times \sum_{j=2}^{n}(T_j - 1) + C5 \times \sum_{j=2}^{n} T_j\prime$

$= C1 \times n + (C2 + C6) \times (n-1) + C3 \times \frac{(2+n)(n-1)}{2} + (C4 + C5) \times \frac{(1+n-1)(n-1)}{2}$

$= k_1 n^2 + k_2 n + k_3$

(b) Repeat part (a) for the *best* case running time.

Best case: when number is ordered from small to large(no C5) $T_j\prime = 0$

$C1 \times n + (C2 + C6) \times (n-1) + C3 \times \sum_{j=2}^{n} T_j + C4 \times \sum_{j=2}^{n}(T_j - 1) + C5 \times 0$

$= C1 \times n + (C2 + C6) \times (n-1) + C3 \times \frac{(2+n)(n-1)}{2} + C4 \times \frac{(1+n-1)(n-1)}{2}$

$= k_4 n^2 + k_5 n + k_6$

(c) Use the O-notation to compare the worst-case and best-case running times computed above to the following functions n, $n \lg n$, and n^2

Worst case = O(n^2) Best case = O(n^2)

(d) Compare the *worst* and *best* case running times of the selection sort to the corresponding times of the insertion sort using one of the three notations, Θ, o, or ω.

	Selection Sort	Insertion Sort
Worst case:	$k_1n^2 + k_2n + k_3$	$a_1n^2 + a_2n + a_3$
Best case:	$k_4n^2 + k_5n + k_6$	$a_1n + a_2$

Therefore, Worst case: $T_s = \Theta(T_i)$ Best case: $T_s = o(T_i)$

2. Use L'Hôpital's theorem to prove that:

$$log(n)^{k_1} = o(n^{k_2})$$

For any values of k_1 and k_2 including the case where k_1 is not integer.

$log(n)^{k_1} = o(n^{k_2})$ is equivalent to $\lim_{n\to\infty} \frac{log(n)^{k_1}}{n^{k_2}} = 0$
Because when n approach infinity, both $log(n)$ and n^{k_2} approach infinity, we can apply L'Hôpital's theorem:

$k_1 \times \lim_{n\to\infty} \frac{\frac{1}{n}}{k_2 \times n^{k_2-1}} = \frac{k_1}{k_2} \times \lim_{n\to\infty} \frac{1}{n^{k_2}} = 0$

Have proved that $\lim_{n\to\infty} \frac{log(n)^{k_1}}{n^{k_2}} = 0 \Rightarrow \mathrm{log(n)}^{k_1} = o(n^{k_2})$

3. Rank the following functions by order of growth; that is, find an arrangement $g_1, g_2, ...$ of the functions staisfying $g_1 = \Omega(g_2), g_2 = \Omega(g_3), ...$ Partition your list ino equivalence classes such that functions $f(n)$ and $g(n)$ are in the same class if and only if $f(n) = \Theta(g(n))$.

$$(\sqrt{2})^{\lg n} \qquad n^2 \qquad n! \qquad (3/2)^n$$

$$n^3 \qquad \lg^2 n \qquad \lg(n!) \qquad 2^{2^n} \qquad \ln\ln n$$

$$1 \qquad \ln n \qquad e^n \qquad (n+1)! \qquad \sqrt{\lg n}$$

$$n \qquad 2^n \qquad n\lg n \qquad 2^{2^n+1}$$

From high to low growth:
$(n{+}1) = \Omega(n{+}1)!$ $\quad (n{+}1)! = \Omega(e^n)$ $\quad e^n = \Omega(2^{2^n})$ $\quad 2^{2^n} = \Theta(2^{2^n+1})$ $\quad 2^{2^n+1} = \Omega(2^n)$ $\quad 2^n = \Omega((3/2)^n)$ $\quad (3/2)^n = \Omega((\sqrt{2})^{\lg n})$ $\quad (\sqrt{2})^{\lg n} = \Omega(n^3)$ $\quad n^3 = \Omega(n^2)$ $\quad n^2 = \Omega(nlog(n))$ $\quad nlog(n) = \Omega(log^2 n)$ $\quad log^2 n = \Omega(log(n!))$ $\quad log(n!) = \Omega(\ln(\ln(n)))$ $\quad \ln(\ln(n)) = \Omega(\ln(n))$ $\quad \ln(n) = \Omega(\sqrt{\lg n})$ $\quad \sqrt{\lg n} = \Omega(n)$ $\quad n = \Omega(1)$