Closed-book and no calculators are allowed.

1. (4 points) Fill in the blank space.

   A. $\lceil \log 20113011 \rceil = (\_\_\_\_\_)$.

   B. $\lfloor \log 1411778724 \rfloor = (\_\_\_\_\_)$.

2. (6 points) True or false.

   A. $(5n^2 + 3n) = \Omega(n)$. ( )

   B. $(5n^2 + 3n) = \Omega(n^2)$. ( )

   C. $(5n^2 + 3n) = \Omega(n^3)$. ( )

3. (18 points) Multiple choice questions. The sorting/search algorithms are the ones we discussed in class.

   (a) Which of the following is true? ____

   A. $2011n + 3011n \log n = O(n)$.

   B. $\frac{2011n}{\log n} + 3011 \log n = \Omega(n)$.

   C. $3011n + \frac{n \log n}{2011} = O(n)$.

   D. $3011n + \frac{n \log n}{2011} = \Omega(n)$.

   (b) Which of the following is false on two positive functions f and g? ____

   A. if $f = \Theta(g)$, then $g = \Omega(f)$

   B. if $f = \Theta(g)$, then $g = \Theta(f)$

   C. if $f = \Theta(g)$, then $g = O(f)$

   D. none of them.

   (c) Sorting the array $2011, 2022, 11, 11, 2, 30$ with bubble sort. How many major iterations? ____

   A. 3 B. 4 C. 5 D. ≥ 6

   (d) Sorting the array $2011, 2022, 11, 11, 2, 30$ with selection sort. How many major iterations? ____

   A. 3 B. 4 C. 5 D. ≥ 6

   (e) Suppose that chars 'P', 'O', 'L', 'Y', and 'U' are pushed onto an originally empty stack in order. There are five successful `pop` operations, and when a value is popped it is printed out. Which sequence **cannot** be the output? ____

   A. P, O, U, L, Y

   B. P, O, L, Y, U

   C. U, Y, L, O, P

   D. Y, L, O, P, U

   (f) We use binary search to search 2011 in the array

   $$2000, 2001, 2002, \ldots, 2035,$$

   How many elements we need to compare 2011 with? ____

   A. 2. B. 3. C. 4. D. 5.

4. (12 points) Write the insertion sort algorithm in Java.

```
void insertionSort(int[] a) {









}
```

5. (2 points (bonus)) The following algorithm is intended to use binary search to find the first element from an array whose key is equal to `key`. Unfortunately, it contains a small bug. Can you identify and fix it?

```
1  public int yimin(int[] a,int key) {
2     int high=a.length;
3     int low=0;
4
5     while(low<high) {
6        int mid=(high+low)/2;
7        if(a[mid]>=key) {
8           high=mid;
9        }
10       else if(a[mid]<key) {
11          low=mid+1;
12       }
13    }
14    if(a[low]==key)
15       return low;
16    return -1;
17 }
```