Name: ______________________________

For each of the following $T(n)$, write the corresponding Big O time complexity. Some series may require research.

1. (2 points) $T(n) = n^2 + 3n + 2$

   1. ______________

2. (2 points) $T(n) = (n^2 + n)(n^2 + \frac{\pi}{2})$

   2. ______________

3. (2 points) $T(n) = 1 + 2 + 3 + \ldots + n - 1 + n$

   3. ______________

4. (3 points) $T(n) = 1^2 + 2^2 + 3^2 + \ldots + (n-1)^2 + n^2$

   4. ______________

5. (2 points) $T(n) = 10$

   5. ______________

6. (2 points) $T(n) = 10^{100}$

   6. ______________

7. (2 points) $T(n) = n + \log n$

   7. ______________

8. (2 points) $T(n) = 12\log(n) + \frac{n}{2} - 400$

   8. ______________

9. (4 points) $T(n) = (n+1) \cdot \log(n) - n$

   9. ______________

10. (4 points) $T(n) = \frac{n^4 + 3n^2 + 2n}{n}$

    10. ______________

11. (5 points) What is the time complexity to insert or remove an item in the middle of an ArrayList?

12. (5 points) Why?

13. (5 points) What is the **average** time complexity to an item to the end of an ArrayList?

14. (5 points) What is the **worst case** time complexity to an item to the end of an ArrayList?

15. (5 points) Taking this all into account, what situations would an ArrayList be the appropriate data structure for storing your data?

______________ 25 points

```java
public static int[] allEvensUnder(int limit){
        if (limit <= 0){
                return new int[0];
        }
        if (limit < 2){
                return new int[1];
        }
        int[] vals =  new int[(limit+1)/2];
        for(int i  = 0;  i <(limit+ 1)/2  ; i++ ) {
                vals[i] = i*2;
        }
        return vals;
}
```

16. (10 points) What is the **time** complexity of the above algorithm?

17. (5 points) What is the **space** complexity of the above algorithm? In other words, how much space is used up based on the input size?

```java
/*
*  https://rosettacode.org/wiki/Sorting_algorithms/Insertion_sort#Java
*/
public static void insertSort(int[] A){
        for(int i = 1; i < A.length; i++){
                int value = A[i];
                int j = i - 1;
                while(j >= 0 && A[j] > value){
                        A[j + 1] = A[j];
                        j = j - 1;
                }
                A[j + 1] = value;
        }
}
```

18. (10 points) What is the time complexity of the above algorithm?

`bogosort` attempts to sort a list by shuffling the items in the list. If the list is unsorted after shuffling, we continue shuffling the list and checking until it is finally sorted.

19. (5 points) What is the worst case run time for `bogosort`?

20. (5 points) Why?

21. (5 points) What is the average case run time for `bogosort` (Hint: think about a deck of cards)?

22. (10 points) Why?