20. In a red-black tree implementation of the Dictionary ADT, the insertItem, findElement, and removeElement operations run in ___1___, ___log n___, and ___log n___ time respectively.

21. To sort a large sequence of keys that cannot fit in memory, ___Merge Sort___ is the recommended choice.

22. In a real-time scenario where a sort must be completed within a fixed amount of time and the input can fit in memory, then ___heap sort___ is the recommended choice.

23. To sort a sequence of less than fifty keys, ___insertion sort___ is the recommended choice.

For questions 24-33, circle True or False (1 point each). Giving a justification is **optional**.

24. The standard Bucket-sort distributes the keys into buckets, sorts each bucket, then concatenates the buckets. This algorithm runs in linear time no matter how the keys are distributed over the buckets.
A) True
B) False

25. A sorting algorithm is considered in-place if it uses some memory in addition to the memory used by the input sequence, but only a constant amount, i.e., not an amount that increases as n increases.
A) True
B) False

26. In a self-balancing binary search tree, to remove a key-element pair, the *removeElement*(k) method calls *expandExternal*(v)
A) True
B) False

27. In a binary search tree, every internal node has either one or two children.
A) True
B) False

28. The lower bound on sorting by key comparisons is $O(n)$ since we can always do a bucket or radix sort.
A) True If memory is big enough.
B) False

29. All implementations of an unordered dictionary are necessarily inefficient for finding items since the entire dictionary might have to be scanned to find the key.
A) True
B) False

30. In Radix-sort, the key is divided into components and Bucket-sort is run on each component, starting with the most-significant (high order) component down to the least significant component (low order).
A) True
B) False