1. (Radix Sort) Suppose we’d like to use RadixSort on an array of $n$ positive integers, where the maximum value we would expect to see in the array is $n^{100}$.

1.1. If we choose to use a base of $r = 10$ in RadixSort, what would be the Big-O runtime of RadixSort (with $M = n^{100}$)? Please write your Big-O runtime only in terms of $n$.

1.2. If we choose to use a base of $r = 100$ in RadixSort, what would be the Big-O runtime of RadixSort (with $M = n^{100}$)? Please write your Big-O runtime only in terms of $n$.

1.3. If we choose to use a base of $r = n$ in RadixSort, what would be the Big-O runtime of RadixSort (with $M = n^{100}$)? Please write your Big-O runtime only in terms of $n$.

2. (Binary Search Trees)

2.1. If we don’t have any guarantees that a Binary Search Tree is “balanced”, what is the worst case Big-O runtime of performing INSERT on a binary search tree?

2.2. Which one of the Red-Black Tree properties listed in class is the one responsible for ensuring that black nodes are roughly balanced throughout the entire RB-Tree? (When we say “roughly balanced”, we refer to the fact that the black nodes would be distributed pretty evenly across the different root-NIL paths in the tree).

   Please write out the description of the property (i.e. describe it using words), rather than just listing the number of the property.

3. (Hash Functions) Indicate whether the following statements are True or False.

3.1. Ideally, we want to use a hash function that guarantees the following: after hashing any $n$ items, the expected size of any bucket is constant (i.e. $O(1)$). True or False?

3.2. In order to store a collection of elements in a hash table, we need to first sort the elements. True or False?

3.3. For a universe of items $U$ and integer $n$, the family $F$ of all hash functions

   \[ h : U \to \{1, n\} \]

   is a universal hash family. True or False?