31. In a Red-Black tree, the restructuring and recoloring operations are sometimes necessary when searching the tree.
A) True
B) False

32. Quick-sort is an example of the divide-and-conquer approach with worst-case time complexity that is no better than Selection-sort or Insertion-sort.
A) True
B) False

33. In a heap, external nodes cannot appear on more than one level.
A) True
B) False

34. [5 points] You are given an algorithm A with running time $\Theta(n^{13})$ in every case (best case, worst case, and average case) and algorithm B with running time $\Theta(n)$ in every case. Very briefly describe why A might be faster than B on some inputs. Your answer **cannot** have anything to do with memory limitations, disk accesses, or paging (i.e., memory is unbounded, disk accesses take 1 unit of time, etc.).

when there is small inputs and if the algorithm is in-place large constants in alg. B.

Algorithm Design:

35. [15 points] (a) Given a sequence of all books in a library (containing title, author, call number, and publisher) and another Sequence of 30 publishers, design an efficient algorithm to determine how many of the books were published by each publisher.

(b) [5 points] What is the time complexity of your algorithm? Justify your answer.

36. (a) [20 points] Given a Sequence B of thousands of credit card bills and another Sequence P of thousands of payments, design an efficient algorithm to create a Sequence of credit card bills that were not paid in full. The elements of Sequence B contain the credit card number, amount due, name, and address. The elements of Sequence P contain the credit card number, amount paid, and name. The output of the algorithm should be a newly created Sequence containing the unpaid bills, i.e., elements with the credit card number, name, address, amount due, and amount paid for those customers for which the amount paid is less than the amount due. Note that you must handle the case where there is a bill, but there is no payment.

(b) [5 points] What is the time complexity of your algorithm? Justify your answer.

37. [20] Design an efficient algorithm to calculate the height and the balance factor of each internal node of a binary tree. The balance factor of an internal node is the height of the left subtree minus the height of the right subtree. Use the methods of the Tree and Binary Tree ADTs to traverse the tree. To set the height and balance factor of each internal node, use methods ***setHeight***(v, h) and ***setBalFactor***(v, bf) where v is a node of the tree, h is the height calculated