These are some review questions to test your understanding of the material. Some of these questions may appear on an exam.

1 B-Trees

1.1 Define B-Tree.

1.2 Give pseudo-code for search in a B-Tree of order M.

1.3 Given a drawing of a B-Tree, show the tree after insertion of a given element.

1.4 Given a drawing of a B-Tree, show the tree after deletion of a given element.

1.5 Draw a B-tree with M = 4 and L = 3 containing the integer values 1 to 25.

1.6 Show the result of inserting the elements `1,3,7,9,5,11,13,6` into an initially empty B-tree having $M = 3$ and $L = 3$. Show the tree at the end of *each* insertion. Assume the key of an element is equal to the element. By definition, M is the order of the B-tree and L is the maximum number of elements that can be stored in a leaf node.

1.7 Given some characteristics of an external storage problem:

1. The number of items to be stored.
2. The size (in bytes) of the key for each item.
3. The size (in bytes) of each item.
4. The block size (in bytes) of a disk block.

design a suitable B-tree (give its order, M and leaf size, L).

1.8 The average-case asymptotic performance of the dictionary operations `find`, `insert`, and `delete` is in $O(\log_2 N)$ for balanced binary-search trees such as Red-Black trees. In a B-tree, the average asymptotic performance of the dictionary operations is in $O(\log_M N)$, where M is the order of the B-tree. Discuss the following:

1. When $M = 2$, do the B-tree and the Red-Black tree have equivalent asymptotic performance for the dictionary operations? Are there advantages of one over the other?
2. B-tree height is proportional to $\log_M N$, indicating that for a given N, a B-tree of high order will be shorter than one of lower order. Is this true? If so, why not always choose a very high value for M since the average asymptotic performance of the dictionary operations is in $O(H)$ (H being the height of the tree)?
3. B-trees find their greatest utility when the data are stored externally (on disk, rather than in RAM). Why is this so?