Problem 1 Hash Functions I \([5+4+4=13 \text{ marks}]\)

Assume that we have a hash table of size \(M = 5\), we use the hash function \(h(k) = k \mod 5\), and we use chaining for collision resolution. Furthermore, assume that our universe is \(U = \{0, 1, 2, \ldots , 12\}\).

a) Demonstrate the insertion of the keys 0, 1, 2, \cdots , 12 into the (initially empty) hash table (in that order). You just need to draw the state of the hash table after all insertions are done.

b) Is the uniform hashing assumption true for this example? Why or why not?

c) [bonus] Suppose only two insertions of elements in \(U\) are made into the initially-empty hash table. If each pair of elements in \(U\) is equally likely to be inserted, show that the probability that the second insertion caused a collision is 11/78. Put a different way, given distinct \(k_1\) and \(k_2\) uniformly chosen from \(U\), show that the probability that \(h(k_1) = h(k_2)\) is 11/78.
Problem 2 Hash Functions II [6 marks]
Assume a hash scheme in which keys are selected uniformly at random from the Universe set \( U = \{1, 2, 3, \ldots, 600\} \). Consider the following two hash functions: \( h_1(k) = k \mod 6 \) and \( h_2(k) = 3k \mod 6 \). Which hash function is better? Justify your answer.

Problem 3 Hash Functions III [6+6=12 marks]
Consider a hash table dictionary with universe \( U = \{0, 1, 2, \ldots, 24\} \) and size \( M = 5 \). If items with keys \( k = 21, 3, 16, 1 \) are inserted in that order, draw the resulting hash table if we resolve collisions using:

a) Linear probing with \( h(k) = (k + 1) \mod 5 \)

b) Cuckoo hashing with \( h_1(k) = k \mod 5 \) and \( h_2(k) = \lfloor k/5 \rfloor \)

Problem 4 Binary Tree Traversals [5+5+5 = 15 marks]

a) Assume you are given the pre-order and in-order traversals of a binary tree \( T_1 \). Note that \( T_1 \) is not necessarily a BST. Explain how we can reconstruct \( T_1 \) given these two traversals.

b) Assume you are given the pre-order and post-order traversal of a binary tree \( T_2 \). Note that \( T_2 \) is not necessarily a BST. Provide an example that shows \( T_2 \) cannot be reconstructed using these traversals. For that, you need to show two different trees with the same pre- and post-order traversals.

c) Assume \( T_3 \) is a binary search tree and its post-order traversal is given. Can we reconstruct \( T_3 \) from this traversal? justify your answer.

Problem 5 BST operations [4+4+6 = 14 marks]
This problem will concern operations on the binary search tree shown in the following figure.

![Binary Tree Diagram]

a) Indicate what nodes are visited when we apply operation \texttt{search(19)}

a) Draw the tree after performing operations \texttt{insert(49)} and then \texttt{insert(13)}.

b) Draw the original tree after performing operations \texttt{delete(9)} and then \texttt{delete(50)}. 
Problem 6  Insertion in BSTs [5 marks]
We saw in the class that rotations can be used to keep a tree balanced (short and fat). A clever student named Bob claims that it is possible to design a binary search tree that balances itself in a way that inserting any item takes time asymptotically less than \( \log n \). Prove that Bob’s claim is wrong. 
Hint: use the lower bound for comparison-based sorting.

Problem 7  2-3 Trees [5+6+6=17 marks]
This problem will concern operations on the 2-3 tree \( T \) shown in the following figure.

![2-3 Tree Diagram]

a) Draw the tree after performing operations insert(49) and insert(5).

b) Draw the original tree after performing operations delete(76) and then delete(55).
See the following figure.

![Modified 2-3 Tree Diagram]

c) Consider an arbitrary 2-3 tree of height \( h \). Provide exact upper and lower bounds for the number of KVPs in such a tree (as a function of \( h \))? Justify your answer.