**DATA STRUCTURES AND ALGORITHMS**

**ASSIGNMENT 03**

**QUESTION 01:**

1. One solution would be to use a dictionary where the keys are the destination, and the value is a list of corresponding chains. Any dictionary implementation would work.

Alternatively, we could modify a separate chaining hash table so it has a capacity of exactly 52 and does not perform resizing. If we then make sure each destination maps to a unique integer from 0 to 51 and hash each train based on this number, we could fill the table and simply print out the contents of each bucket.

A third solution would be to use a BST or Red-Black tree and have each train be compared first based on destination then based on their other attributes. Once we insert the trains into a tree, we can print out the trains sorted by destination by doing an in-order traversal.

1. Regardless of which solution we modify, we would first need to ensure that the train objects are compared first by destination, and second by departure time.

We can modify our first solution by having the dictionary use a sorted set for the value, instead of a list. (The sorted set would be implemented using a BST or a Red-Black tree).

We can modify our second solution in a similar way by using specifically a BST or a Red-Black tree as the bucket type.

Our third solution actually requires no modification: if the trains are now compared first by destination and second by departure time, the Red-Black and BST tree’s iterator will naturally print out the trains in the desired order.

**QUESTION** **02**:

**STRATEGY**

The data structure used for the implementation of above guessing game is **BINARY TREE** in which we have two nodes of each root.

**EXPLANATION**

* There will be ten roots which will hold each question.
* Each node will have two nodes having answers YES & NO, as binary tree only have two child nodes.
* If the answer of particular question results in YES it will not proceed further.
* But if last question will be again a NO then the game will end, as there is no further node in tree.

