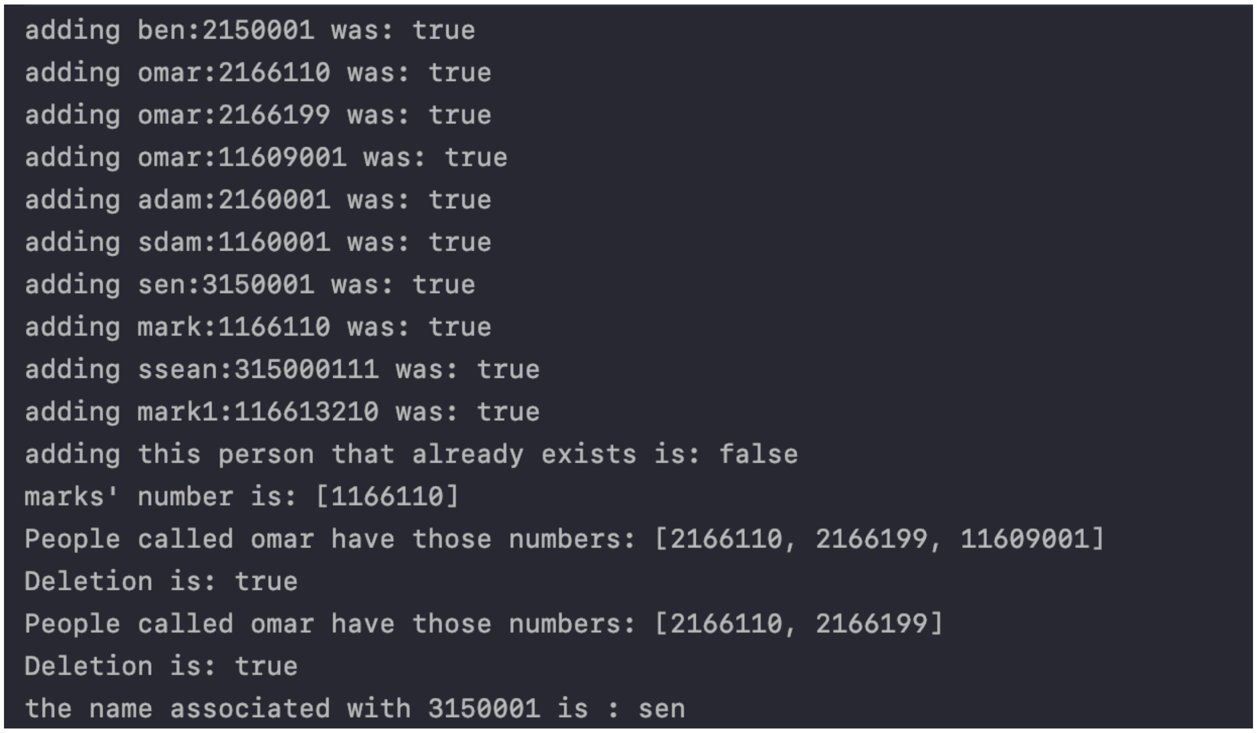
Assignment 3 lab report:



```
adding ben:2150001 was: true
adding omar:2166110 was: true
adding omar:2166199 was: true
adding omar:11609001 was: true
adding adam:2160001 was: true
adding sdam:1160001 was: true
adding sen:3150001 was: true
adding mark:1166110 was: true
adding ssean:315000111 was: true
adding mark1:116613210 was: true
adding this person that already exists is: false
marks' number is: [1166110]
People called omar have those numbers: [2166110, 2166199, 11609001]
Deletion is: true
People called omar have those numbers: [2166110, 2166199]
Deletion is: true
the name associated with 3150001 is : sen
```

This was the snapshot that resulted when adding many numbers and names to the list. Some of them existed but had different numbers and one that had the same number and name and it wasn't added.
Then I conducted searches on both names that had many numbers and ones that had one.
I tried to delete a number from the linked list of numbers and it worked, deleting a nod with children also works as well as a nod without children.
Searching using number search also works.

Added classes were two node classes one that represented the node on the name tree and was allowed to have more than one number for a name so it stored it in a linked list.
The other added class is a node class that has a number and name field that were both strings.

Method runtimes:
PhBInsert runs at 2*log(n), n being the number of elements in the tree, because it adds the element to two trees. Average case: O(logn) worst case is O(n) if the tree only has one branch

PhBDelete runs at 2*log(n), n being the number of elements in the tree, because it deletes the element from two trees. Average O(logn) worst case is O(n) if the tree only has one branch

PhBNameSearch runs at log(n) as it searches through a BST. Average case: O(logn) worst case is O(n) if the tree only has one branch

PhBPhoneSearch also runs at O(logn) in the average case if the tree is balanced, worst case is O(n)

Hidden methods:
numInsert runs at Average case: O(logn) worst case is O(n) if the tree only has one branch

nameInsert runs at k * log(n), n is the number of elements in the tree and k is the number of phone numbers in the linked list. But k should not be a big number and the runtime should be treated as if k was a small constant so average case: O(logn) worst case is O(n) if the tree only has one branch

nameToDelete runs at k * log(n), n is the number of elements in the tree and k is the number of phone numbers in the linked list. And this method calls another method (deleteByName that is constant) But k should not be a big number and the runtime should be treated as if k was a small constant. Average case: O(logn) worst case is O(n) if the tree only has one branch

deleteByName should have a constant or very close to constant because the search had already been done and the method only replaces elements. O(1)

deletebynumber is a recursive method and runs on log n time if the binary tree is balanced, however it runs O(n) on its worst case