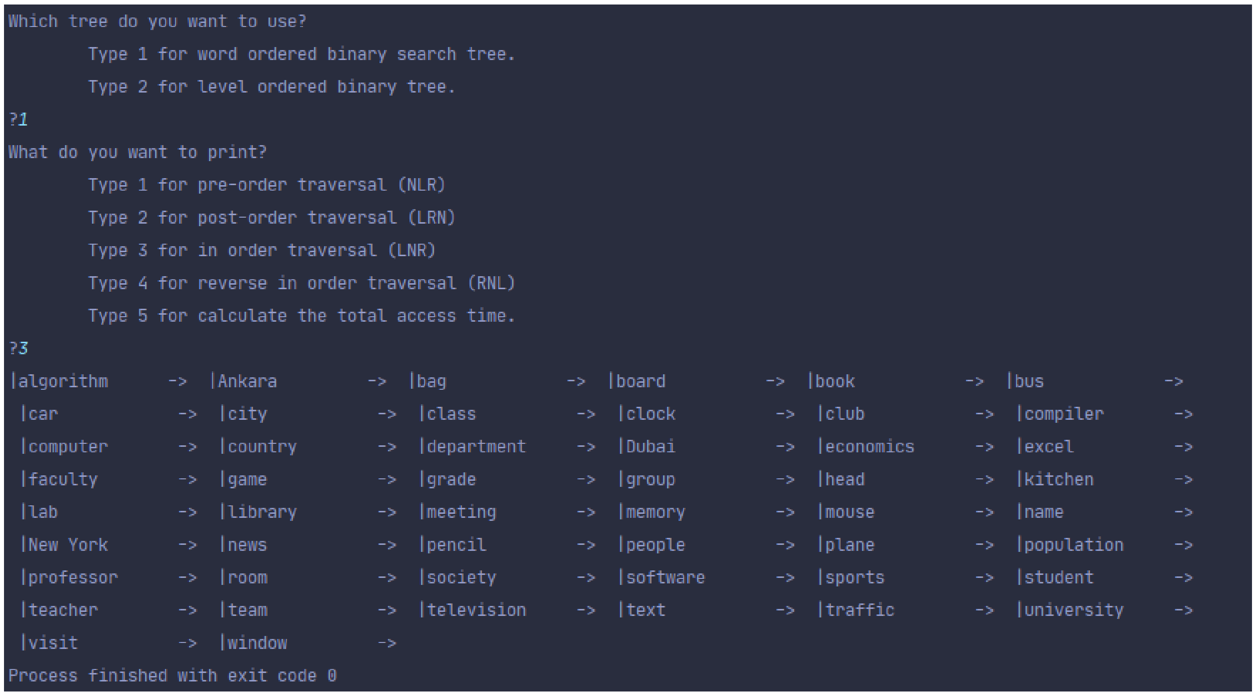
Yasin Tarakçı - 150118055
Report of Project #1
CSE2025 Data Structures

a) Firstly, the program reads the whole file and creates the binary search tree according to given words. The program assigns the words according to their alphabetical order.

This is my program. I added some user experience. And the written list is our binary search tree's in-order traversal.

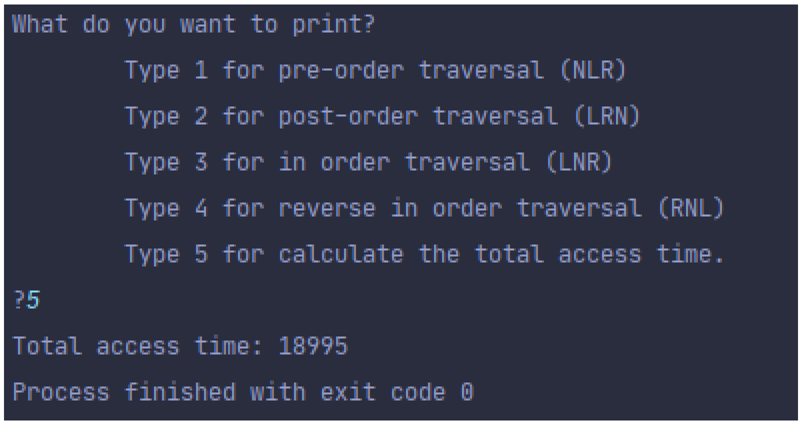


```
Which tree do you want to use?
        Type 1 for word ordered binary search tree.
        Type 2 for level ordered binary tree.
?1
What do you want to print?
        Type 1 for pre-order traversal (NLR)
        Type 2 for post-order traversal (LRN)
        Type 3 for in order traversal (LNR)
        Type 4 for reverse in order traversal (RNL)
        Type 5 for calculate the total access time.
?3
|algorithm     ->  |Ankara        ->  |bag           ->  |board         ->  |book          ->  |bus           ->
 |car          ->  |city          ->  |class         ->  |clock         ->  |club          ->  |compiler      ->
 |computer     ->  |country       ->  |department    ->  |Dubai         ->  |economics     ->  |excel         ->
 |faculty      ->  |game          ->  |grade         ->  |group         ->  |head          ->  |kitchen       ->
 |lab          ->  |library       ->  |meeting       ->  |memory        ->  |mouse         ->  |name          ->
 |New York     ->  |news          ->  |pencil        ->  |people        ->  |plane         ->  |population    ->
 |professor    ->  |room          ->  |society       ->  |software      ->  |sports        ->  |student       ->
 |teacher      ->  |team          ->  |television    ->  |text          ->  |traffic       ->  |university    ->
 |visit        ->  |window        ->
Process finished with exit code 0
```

b) I also calculated the total access time with my program. The total access time for our binary search tree is 18995. I calculated it with using each node's depth and frequency.

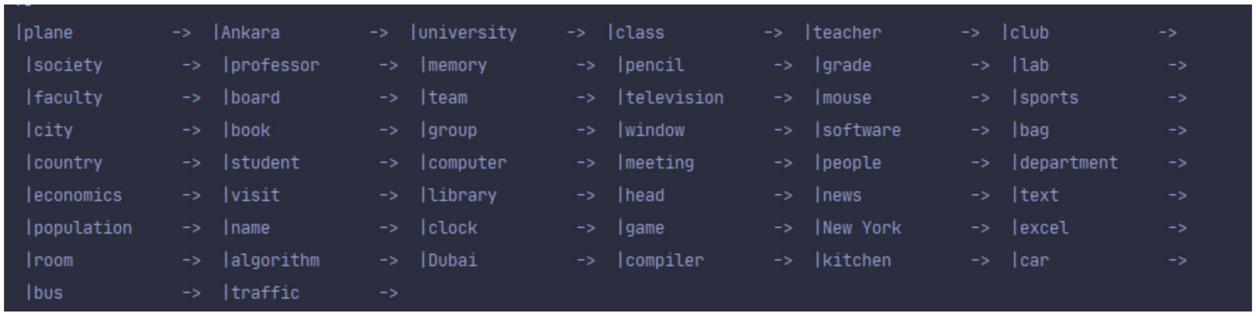
$$Total\ access\ time \mathrel{+}= (depth + 1) * frequency$$

I use this formula for each node one by one. This is the output of my program.



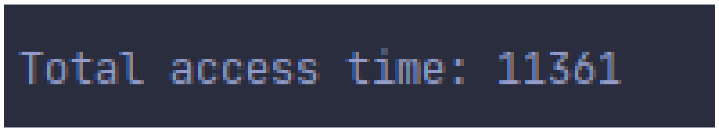
```
What do you want to print?
        Type 1 for pre-order traversal (NLR)
        Type 2 for post-order traversal (LRN)
        Type 3 for in order traversal (LNR)
        Type 4 for reverse in order traversal (RNL)
        Type 5 for calculate the total access time.
?5
Total access time: 18995
Process finished with exit code 0
```

c) For the second part of my program, I re-open the file and read it again line by line. But this time I create the nodes and assign them to the node array. I assign them an array because I will sort the nodes according to their frequencies. When I sort the array, I can start creating the binary tree. I insert the nodes into the tree in level order. Nodes with the highest frequency are placed closer to the root. By doing this I reduced the total access time.
This is the in-order traversal for the balanced binary tree.



```
|plane         ->  |Ankara        ->  |university    ->  |class         ->  |teacher       ->  |club          ->
 |society       ->  |professor     ->  |memory        ->  |pencil        ->  |grade         ->  |lab           ->
 |faculty       ->  |board         ->  |team          ->  |television    ->  |mouse         ->  |sports        ->
 |city          ->  |book          ->  |group         ->  |window        ->  |software      ->  |bag           ->
 |country       ->  |student       ->  |computer      ->  |meeting       ->  |people        ->  |department    ->
 |economics     ->  |visit         ->  |library       ->  |head          ->  |news          ->  |text          ->
 |population    ->  |name          ->  |clock         ->  |game          ->  |New York      ->  |excel         ->
 |room          ->  |algorithm     ->  |Dubai         ->  |compiler      ->  |kitchen       ->  |car           ->
 |bus           ->  |traffic       ->
```

d) I calculated the total access time with my program. The total access time for my binary tree is 11361. I used the same formula which I used in b part.



```
Total access time: 11361
```

e) The calculated total access times are different for my trees. The first BST is using the words for key and the nodes insert in the line order to tree. In other words, the BST is not bounded to the frequency. But the second tree, firstly I ordered the nodes in order to their frequency and insert them to the tree in level order. Nodes with the highest frequency are placed closer to the root for this reason total access time of the second tree is smaller than the first tree.