ASSIGNMENT - 2 EXTERNAL DOCUMENT

Overview

The assignment 2 was divided into two problems. The task of **problem 1** was to write test cases for the code. I had to write test cases for the **recommend()** method which will recommend products to user may be interested to buy with respect to products that are currently in the shopping cart. The **recommend()** method will tally the record of the past purchase records with the products in shopping cart and will return at most 5 product.

The task of **problem 2** was to implement a data structure that looks like a linked list but exhibits properties of balanced binary tree. To achieve this, I had to write a code that will add a value to the linked list and find if the value is already present in the linked list or not.

add() : For adding the value inside the linked list, we will first insert the value in the lowest list and will keep the list sorted. Next, a coin will be tossed, and it will randomly choose "1" or "0". If "1" appears on coin then we need to insert the value in upper list, else if "0" appears then we will add the value in the lower list.

find() : Find method will search inside the linked list if the value is in the list or not. It will return true if the value is found or else it will return false.

Solution

In this assignment I solved the whole problem using doubly linked list. The reason I chose doubly linked list is that it helps to point the nodes and keep track of the flow. In addition, it can traverse both the direction and insert and find a node becomes easy.

How I implemented

1). I created a class by name **DoublyLLNode** which will be used to create all the nodes that are needed to point nodes, create a new node and traverse through nodes in the doubly linked list.

2). Next, I created another class by name **DoublyLL** in which all the methods are present and is used to initialize reference nodes to null.

3). Inside class **DoublyLL**, I created a method name **MyAdd(),** which accepts string value from user and returns a boolean value. Further, a new node will be created for the value that user wants to insert.

4). Next, it will check if the value is null or is empty, if it is the case then it will return false.

5). In next step it will check if head is null then, it inserts the value into the list and the value that was just added becomes the head, as it is the very first node in the doubly linked list.

6). In next else if statement, it will check if the value user wants to insert is already present in the linked list or not. For checking purpose I called the **MyFind()** method which will return true if it exists or return false if it does not.

7). Further in else part, where the actual value will be added. To add the value we need to go to the last node and for that I used a while loop which execute till mypresent.next ! = null and inside the while loop we will traverse each node. And once we find the last node, we will attach last nodes next to the new node and new nodes previous to last node. Lastly I called a **sortList()** method which will sort the nodes in ascending order.

8). After flipping the coin, if the coin generates value as "1" then a new node will be created and it will check the first level list whether the head's up is not null if it so, then the new node will be added to upper level and kept connection between the nodes of the two lists . And if the coins generate value as "0" then it will not go to upper level.

9). In **sortList()** method I am passing the complete head value as a parameter and is storing it in a new node i.e. present. Present will store the first value of linked list and another node ahead will store the next value of present. Further, I had used bubble sort to sort the values.

10). **MyFind()** method accepts one parameter from user that is needed to search and will return boolean value. Firstly, it checks whether head value is null or not, if it is then it will return false. Else if it will check whether the searched value is null or not, if it is then it will again return false. Else, I created a new node that stores the value of head. Next, I used a while loop to traverse till it does not find null and inside the while loop, I am checking if the search value matches with any value in the linked list, if it matched then it will return true or else false.

What I could not implement in this assignment.

I implemented all the parts of the problem 2, but the only part I could not implement was to add as many levels as we want. I was able to achieve two levels of hierarchy that operate correctly with add and find method. I got the flow of how to implement more than two levels, however I was unable to write the code.