1. Write a function create_binary_tree() that creates a binary tree from a given array. [3]

The TreeNode class has been defined for you:

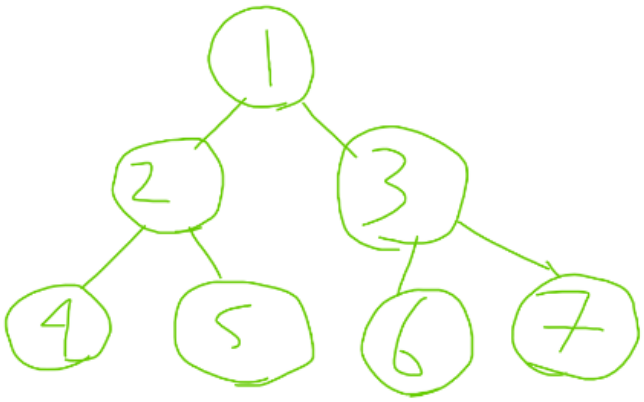
```
class TreeNode:
    def __init__(self, value):
        self.value = value
        self.left = None
        self.right = None
```

Example:

Input array:

```
n = [1, 2, 3, 4, 5, 6, 7]
```

Output Tree:



2. Write the insert() function and __hash_function() of the HashTable class which uses an array to store a key-value pair, where the key is a string representing a fruit, and the value is an int representing its respective price. [7]

__hash_function(key)

Takes a key which is a string, and calculates its length. If length is even, the sum of the ascii values of the even characters is calculated, otherwise, the sum of the ascii values of odd characters is calculated. Finally, it returns the sum modded by length of the array

If we call __hash_function("apple"):
As len("apple") is odd, characters in odd indexes (p, l) are taken:

sum = ord(p) + ord(l) = 112 + 108 = 220

If the Hashtable object is initialized with length 5, then:

sum % length of Hashtable array = 220 % 5 = 0

So, the function returns 0

insert(key, value)
Creates a node that contains a tuple which stores the key-value pair as (key, value). Finds index by passing key to __hash_function(). If there is no collision, the node is placed in the index.

If there is a collision, forward chaining is applied. Here, the chain should be arranged in **descending** order, i.e, if the node being inserted has the highest value (price), it will be the head of the chain. Otherwise, you should iterate the chain and insert it in the appropriate position.