1. Write a program in Java to implement a binary search tree (BST). The program will insert five names into the BST. All names must be read from the console. After insertion, the program will show five names in ascending order. Then it will ask for any name from the user. The name will be searched in the BST, and if found, the name will be deleted from the tree. Otherwise, an appropriate message will be shown. (**Do not use any built-in tree class.**) The sample inputs/ outputs are given below. **(25 points)**

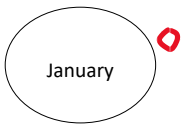
Sample inputs and outputs:
(User's inputs are shown in **bold**) **Sample execution 1:**
Enter 5 names: **Bob Edwin Cory Alice Mark**
The names in the BST: Alice Bob Cory Edwin Mark
Enter the name to be removed: **Edwin**
The name is removed. The elements in the BST: Alice Bob Cory Mark

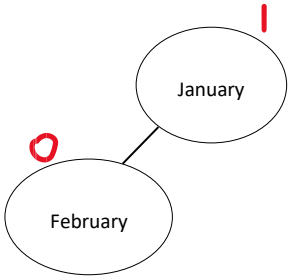
Sample execution 2:
Enter 5 names: **Bob Edwin Cory Alice Mark**
The names in the BST: Alice Bob Cory Edwin Mark
Enter the name to be removed: **Travis** The name is not found.

2. Insert the names of 12 months in an AVL tree in the regular order from January to December. **Treat the month names as dictionary words, NOT as the month's position.** For instance, January as a month comes before February, but the word 'February' comes before 'January' in the English dictionary. Show each step of insertion and rotation. **(25 points)**

For instance, nodes should be inserted as follows:

January





February

and so on.

Bonus question:

3. Represent the following two code snippets as appropriate functions by calculating primitive operations of each statement and hence find the Big-Oh running time complexity: **(20 points)**

a)
```
public static int code1(int[ ] data)
{
 int low = 0, high = data.length − 1;   while
 (low < high)
  {
        int temp = data[low];
        data[low++] = data[high];
        data[high−−] = temp;
  }   int i
 = 0;
  while(i < data.length)
        System.out.println(data[i++]);
}
```

b)
```
public static int code2(int[ ] first, int[ ] second)
{
 int n = first.length, count = 0;
 for(int i=0; i < n; i++)
 {                int
total = 0;
        for(int j=0; j < n; j++)
        for(int k=0; k <= j; k++)
                total += first[k];
        if(second[i] == total)
                count++;
 }
 return count;
}
```

Upload the files (.java for question 1 and .docx for questions 2 and 3) . For the 2nd question, you can draw each step on paper, take photos, and paste those photos into a Word file.