Module - 2 : Assessment

1. Score of a String

```java
import java.util.Scanner;

public class Qn_1 {
    private static int absDiff(int a, int b) {
        if ( a - b <= 0) return (a - b) * - 1;
        else return (a - b);
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the String: ");
        String str = sc.nextLine();
        int prev = 0, res = 0;
        for(int i=0; i<str.length()-1; i++) {
            res += absDiff(str.charAt(i), str.charAt(i+1));
        }
        System.out.println(res);
    }
}
```

Output

```
Enter the String: hello
13
```

---

2. Magical String

```java
package usr.Assessments;

import java.util.Scanner;

public class Qn_2 {
    public static String magicalString(String str) {
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < str.length(); i++) {
            if (sb.length() == 0) {
                sb.append(str.charAt(i));
            } else {
                if (Math.abs(sb.charAt(sb.length() - 1) - str.charAt(i)) == 32) {
                    sb.deleteCharAt(sb.length() - 1);
                } else {
                    sb.append(str.charAt(i));
                }
            }
        }
        return sb.toString();
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String str = sc.nextLine();
        String val = sc.nextLine();
        System.out.println(magicalString(str));
        System.out.println(magicalString(val));
```

```
    }
}
```

Output

```
sweeEet
codeE
sweet
cod
```

---

3. Reverse Alternate k Nodes

```java
package usr.Assessments;

import java.util.Scanner;

public class Qn_3 {
    public static void reverseAlternateKNodes(int[] arr, int k) {
        if (arr == null || arr.length < k) return;
        int temp = arr[0];
        for (int i = 0; i < k; i++ ) {
            arr[i] = arr[i+1];
        }
        arr[k] = temp;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the values of the array: ");
        String[] str = sc.nextLine().split(" ");
        int[] arr = new int[str.length];
        for (int i = 0; i < str.length; i++) {
            arr[i] = Integer.parseInt(str[i]);
        }
        System.out.print("Enter the value of k: ");
        int k = sc.nextInt();
        reverseAlternateKNodes(arr,k-1);
        for (int i: arr) {
            System.out.print(i + " ");
        }
    }
}
```

Output

```
Enter the values of the array:
5 6 7 8 9 10 11 12
Enter the value of k: 3
6 7 5 8 9 10 11 12
```

---

4. Least Number of Unique Integers after K Removals

```java
package usr.Assessments;

import java.util.Scanner;

public class Qn_4 {
    public static int[] countSort(int[] inputArray) {
```

```
        int N = inputArray.length;
        int M = 0;
        for (int j : inputArray) {
            M = Math.max(M, j);
        }
        int[] countArray = new int[M + 1];

        for (int j : inputArray) {
            countArray[j]++;
        }

        for (int i = 1; i <= M; i++) {
            countArray[i] += countArray[i - 1];
        }

        int[] outputArray = new int[N];

        for (int i = N - 1; i >= 0; i--) {
            outputArray[countArray[inputArray[i]] - 1] = inputArray[i];
            countArray[inputArray[i]]--;
        }

        return outputArray;
    }

    public static int leastNumberUniqueIntegersAfterKRemovals(int[] arr, int k) {
        arr = countSort(arr);
        int count = 1;
        for (int i = k; i < arr.length - 1; i++) {
            if (arr[i] != arr[i + 1]) {
                count++;
            }
        }
        return count;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the values of arrays: ");
        String[] str = sc.nextLine().split(" ");
        int[] arr = new int[str.length];
        for (int i = 0; i < str.length; i++) {
            arr[i] = Integer.parseInt(str[i]);
        }
        System.out.println("Enter the values of k: ");
        int k = sc.nextInt();
        System.out.println(leastNumberUniqueIntegersAfterKRemovals(arr,k));
    }
}
```

Output

```
Enter the values of arrays:
4 3 1 1 3 3 2
Enter the values of k:
3
2
```

---

5. All Elements in Two Binary Search Trees

```
import usr.collection.BST;
```

```java
import java.util.*;


public class Qn_5 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the values of BST one: ");
        String[] bst = sc.nextLine().split(" ");
        System.out.println("Enter the values of BST two: ");
        String[] bst2 = sc.nextLine().split(" ");
        BST<Integer> res = new BST<>();
        for(String s : bst) {
            res.insert(Integer.parseInt(s));
        }
        for(String s : bst2) {
            res.insert(Integer.parseInt(s));
        }
        res.getInOrder();
    }
}
```

BST.java

```java
package usr.collection;

import java.util.ArrayList;
import java.util.List;

class BSTNode<E extends Comparable<E>>{
    public E data;
    public BSTNode<E> left;
    public BSTNode<E> right;

    public BSTNode(E data) {
        this.data = data;
        left = null;
        right = null;
    }


    public BSTNode(E data,BSTNode<E> left, BSTNode<E> right) {
        this.data = data;
        this.left = left;
        this.right = right;
    }
}

@SuppressWarnings({"", "ClassEscapesDefinedScope"})
public class BST<E extends Comparable<E>> {
    private BSTNode<E> _root;
    public BST() {
        _root = null;
    }

    public BST(E data, BSTNode<E> left, BSTNode<E> right) {
        _root = new BSTNode<>(data, left, right);
    }

    @SuppressWarnings("ClassEscapesDefinedScope")
    public BST(BSTNode<E> root) {
        _root = root;
    }

    public void insert(E data) {
        BSTNode<E> newNode = new BSTNode<>(data);
        if (_root == null) {
```

```
            _root = newNode;
        } else {
            BSTNode<E> current = _root;
            while (true) {
                if (current.data.compareTo(data) > 0) {
                    if (current.left == null) {
                        current.left = newNode;
                        break;
                    }
                    current = current.left;
                } else {
                    if (current.right == null) {
                        current.right = newNode;
                        break;
                    }
                    current = current.right;
                }
            }
        }
    }


    public void display() {
        BSTNode<E> current = _root;
        System.out.println("Binary Search Tree: ");
        this._printTree(current, 0);
        System.out.println();
    }

    private void _printTree(BSTNode<E> node, int level) {
        if (node != null) {
            _printTree(node.right, level + 1);
            for(int i=0; i<level; i++) {
                System.out.print("      ");
            }
            System.out.println(" -> " + node.data);
            _printTree(node.left, level + 1);
        }
    }

    public void getInOrder() {
        _getInOrder(this._root);
        System.out.println();
    }

    private void _getInOrder(BSTNode<E> node) {
        if (node == null) {
            return;
        }
        _getInOrder(node.left);
        System.out.print(node.data + ", ");
        _getInOrder(node.right);
    }
    public void inorderTraversal(BSTNode<E> root, List<E> inorder) {
        if (root == null) return;
        inorderTraversal(root.left, inorder);
        inorder.add(root.data);
        inorderTraversal(root.right, inorder);
    }

    public BSTNode<E> get_root() {
        return this._root;
    }
}
```

Output

```
Enter the values of BST one:
2 1 4
Enter the values of BST two:
1 0 3
0, 1, 1, 2, 3, 4
```