CASE TOOLS

1. Write a java program to check palindrome number.

```
public class Palindrome {

   public static void main(String[] args) {

       Palindrome palindrome = new Palindrome();

       String input = "Poor Dan is in a droop";

       if (palindrome.isPalindrome(input))
           System.out.println("\"" + input + "\" is a palindrome.");
       else
           System.out.println("\"" + input + "\" is not a
palindrome.");

   }

   boolean isPalindrome(String input) {

       //Converting input to Lowercase and removing all white spaces
       input = input.toLowerCase().replaceAll("\\s+", "");

       //Length of input
       int length = input.length();

       //Check if non alpha-numeric string
       if (!input.matches("^([a-z0-9]+)?$")) {
           return false;
       }

       //Character by character comparision
       for (int i = 0; i < length / 2; i++) {
           if (input.charAt(i) != input.charAt(length - i - 1)) {
               return false;
           }
       }
       return true;
```

```
    }

}
```

```
import org.junit.After;
import org.junit.Before;
import org.junit.Test;

import static org.junit.Assert.*;

public class PalindromeUnitTest {

    private Palindrome palindrome;
    private String input;

    @Before
    public void setUp() throws Exception {

        input = null;
        palindrome = new Palindrome();

    }

    @After
    public void tearDown() throws Exception {

    }

    @Test(expected = NullPointerException.class)
    public void nullStringTest() throws Exception {

        palindrome.isPalindrome(null);

    }

    @Test
    public void emptyStringTest() throws Exception {

        input = "";
```

```
        assertTrue(palindrome.isPalindrome(input));

    }

    @Test
    public void multipleWhiteSpaceTest() throws Exception {

        input = "A   Santa         at Nasa";

        assertTrue(palindrome.isPalindrome(input));

    }

    @Test
    public void singleCharTest() throws Exception {

        input = "H";

        assertTrue(palindrome.isPalindrome(input));

    }

    @Test
    public void punctuationTest() throws Exception {

        input = "Eva, can I see bees in a cave?";

        assertFalse(palindrome.isPalindrome(input));

    }

    @Test
    public void unicodeTest() throws Exception {

        input = "\u20A9 My gym \u20A9";

        assertFalse(palindrome.isPalindrome(input));

    }
```

```
    @Test
    public void alphaNumericPalindromeTest() throws Exception {

        input = "Air 2 an a2ria";

        assertTrue(palindrome.isPalindrome(input));
    }

    @Test
    public void validPalindromeTest() throws Exception {

        input = "No lemon no melon";

        assertTrue(palindrome.isPalindrome(input));
    }

    @Test
    public void invalidPalindromeTest() throws Exception {

        input = "I am a tester";

        assertFalse(palindrome.isPalindrome(input));
    }
}
```

2. Write a java program to check Armstrong number.

```
public class Armstrong {

    public boolean isValid(int candidate) {
        char[] digits = valueOf(candidate).toCharArray();
        int result = 0;
        for (char digit : digits) {
            int d = Character.digit(digit, 10);
            result += Math.pow(d, digits.length);
        }
        return result == candidate;
    }
}
```

```java
import org.junit.Test;

import static org.fest.assertions.Assertions.assertThat;


public class ArmstrongUnitTest {

   @Test
   public void _153IsValidThreeDigitArmstrongNumber() {
       boolean valid = new ArmstrongValidator().isValid(153);

       assertThat(valid).isTrue();
   }

   @Test
   public void _1634IsValidFourDigitArmstrongNumber() {
       boolean valid = new ArmstrongValidator().isValid(1634);

       assertThat(valid).isTrue();
   }

   @Test
   public void _370IsValidThreeDigitArmstrongNumber() {
       boolean valid = new ArmstrongValidator().isValid(370);

       assertThat(valid).isTrue();
   }

   @Test
   public void _371IsValidThreeDigitArmstrongNumber() {
       boolean valid = new ArmstrongValidator().isValid(371);

       assertThat(valid).isTrue();
   }

   @Test
   public void _407IsValidThreeDigitArmstrongNumber() {
       boolean valid = new ArmstrongValidator().isValid(407);
```

```
        assertThat(valid).isTrue();
    }

    @Test
    public void _8208IsValidFourDigitArmstrongNumber() {
        boolean valid = new ArmstrongValidator().isValid(8208);

        assertThat(valid).isTrue();
    }
}
```

3. Write a java program to sort an array element using bubble sort algorithm.

```
import java.util.stream.IntStream;

public class BubbleSort {

    void bubbleSort(Integer[] arr) {
        int n = arr.length;
        IntStream.range(0, n - 1)
          .flatMap(i -> IntStream.range(1, n - i))
          .forEach(j -> {
              if (arr[j - 1] > arr[j]) {
                  int temp = arr[j];
                  arr[j] = arr[j - 1];
                  arr[j - 1] = temp;
              }
          });
    }
}
```

```
public class BubbleSortUnitTesting {
    @Test
    public void
givenIntegerArray_whenSortedWithBubbleSort_thenGetSortedArray() {
        Integer[] array = { 2, 1, 4, 6, 3, 5 };
        Integer[] sortedArray = { 1, 2, 3, 4, 5, 6 };
        BubbleSort bubbleSort = new BubbleSort();
        bubbleSort.bubbleSort(array);
        assertArrayEquals(array, sortedArray);
```

```
    }
```

4. Write a java program to sort an array element using insertion sort algorithm.

```
package logic;

public class InsertionSort {
   public static void insertionSort(int[] input) {
   for (int i = 1; i < input.length; i++) {
        int key = input[i];
        int j = i - 1;
        while (j >= 0 && input[j] > key) {
            input[j + 1] = input[j];
            j = j - 1;
        }
        input[j + 1] = key;
   }
}
}
```

```
package testcases;

import logic.*;
import org.junit.Test;
import static org.junit.Assert.assertArrayEquals;



public class InsertionSortUnitTest {

   @Test
   public void InsertionSortSortedAsc() {
        int[] input = {6, 2, 3, 4, 5, 1};
        InsertionSort.insertionSort(input);
        int[] expected = {1, 2, 3, 4, 5, 6};
        assertArrayEquals("the two arrays are not equal", expected, input);
   }
}
```

5. Write a java program to perform binary search in java.

```
import java.util.Arrays;
```

```java
import java.util.Collections;
import java.util.List;

public class BinarySearch {

   public int runBinarySearchIteratively(int[] sortedArray, int key,
int low, int high) {

       int index = Integer.MAX_VALUE;

       while (low <= high) {

           int mid = low + ((high - low) / 2);

           if (sortedArray[mid] < key) {
               low = mid + 1;
           } else if (sortedArray[mid] > key) {
               high = mid - 1;
           } else if (sortedArray[mid] == key) {
               index = mid;
               break;
           }
       }
       return index;
   }

}


import java.util.Arrays;
import java.util.List;
import org.junit.Assert;
import org.junit.Test;

public class BinarySearchUnitTest {

   int[] sortedArray = { 0, 1, 2, 3, 4, 5, 5, 6, 7, 8, 9, 9 };
   int key = 6;
   int expectedIndexForSearchKey = 7;
   int low = 0;
```

```
    int high = sortedArray.length - 1;
    List<Integer> sortedList = Arrays.asList(0, 1, 2, 3, 4, 5, 5, 6,
7, 8, 9, 9);

    @Test
    public void
givenASortedArrayOfIntegers_whenBinarySearchRunIterativelyForANumber
_thenGetIndexOfTheNumber() {
        BinarySearch binSearch = new BinarySearch();
        Assert.assertEquals(expectedIndexForSearchKey,
binSearch.runBinarySearchIteratively(sortedArray, key, low, high));
    }

}
```

6. Write a java program to find 2nd largest number in an array.

```
public class Seconglargestarray {
public static int print2largest(int arr[],int arr_size)
{
 int i, first, second;
 // There should be
 // atleast two elements
 if (arr_size < 2)
 {
   System.out.printf(" Invalid Input ");
   return second;
 }
 // Sort the array
 Arrays.sort(arr);
 // Start from second last element
 // as the largest element is at last
 for (i = arr_size - 2; i >= 0; i--)
 {
   // If the element is not
   // equal to largest element
   if (arr[i] != arr[arr_size - 1])
   {
     return arr[i];
```

```
    }
 }
 return -1;
}
}
```

7. Write a java program to transpose a matrix.

```
public class Transpose {
   public static int[][] transpose(int A[][], int B[][])
   {
       int i, j;
       for (i = 0; i < N; i++)
           for (j = 0; j < N; j++)
               B[i][j] = A[j][i];

   return B;
}
}
```

8. Write a java program to remove duplicate element in an array.

```
9.
10.
11. import java.util.*;
12. public class DuplicateinArray {
13.
14.     public List<String> removeDups(String[] str){
15.         List<String> strList = new ArrayList<>();
16.         int count = 0;
17.         for(int i = 0; i < str.length; i++){
18.             for(int j = i+1; j < str.length; j++){
19.                 if(str[i].equals(str[j])){
20.                     count += 1;
21.                 }
22.             }
23.             if(count < 1){
24.                 strList.add(str[i]);
```

```
            }
            count = 0;
        }
        for(int k = 0; k < strList.size(); ){
            System.out.println("check "+strList);
            System.out.println(strList.getClass());
            return(strList);

        }
        return null;
    }
}
```

```
import static org.junit.Assert.*;

import java.util.Arrays;

import org.junit.Test;

public class DuplicateinArrayUnitTest {
   public void test() {

       DuplicateinArray rd = new DuplicateinArray();

       String[] strArray =
{"yellow","blob","blob","yellow","talk","chat","yellow","talk"};
       List<String> checkList = rd.removeDups(strArray);


assertEquals(Arrays.asList("yellow","blob","talk","chat").tostring(),check
List.toString());
   }
}
```

37. Write a java program to GCD of given two numbers.

```
public class GCD {
   public static int gcdByEuclidsAlgorithm(int n1, int n2) {
       if (n2 == 0) {
           return n1;
```

```
        }
        return gcdByEuclidsAlgorithm(n2, n1 % n2);
    }
}
```

38. Implement the suduko application.
https://github.com/prilily/Sudoku-