**<https://github.com/maxiplux/MUM-FPP390-W1L4-HOMEWORK-ASSIGNMENT>**

**package** problem1;

**import** java.util.ArrayList;

**public** **class** Solution1 {

**public** **static** String[] PivotHelper(String[] array, String pivot, Boolean less) {

ArrayList<String> temp = **new** ArrayList<String>();

**if** (array.length <= 1 || array == **null**) {

**return** temp.toArray(**new** String[temp.size()]);

}

**for** (**int** i = 0; i < array.length; i++) {

**if** (less) {

**if** (array[i].hashCode() < pivot.hashCode())

{

temp.add(array[i]);

}

}

**else** {

**if** (array[i].hashCode() > pivot.hashCode())

{

temp.add(array[i]);

}

}

}

**return** temp.toArray(**new** String[temp.size()]);

}

**public** **static** String[] JoinHelper(String[] left, String pivot, String[] right) {

ArrayList<String> temp = **new** ArrayList<String>();

**for** (**int** i = 0; i < left.length; i++)

{

temp.add(left[i]);

}

temp.add(pivot);

**for** (**int** i = 0; i < right.length; i++) {

temp.add(right[i]);

}

**return** temp.toArray(**new** String[temp.size()]);

}

**public** **static** String[] quickSort(String[] array) {

**if** (array.length <= 1) {

**return** array;

}

String pivot = array[0];

String[] left = *quickSort*(*PivotHelper*(array, pivot, **true**));

String[] right = *quickSort*(*PivotHelper*(array, pivot, **false**));

**return** *JoinHelper*(left, pivot, right);

}

**public** **static** **void** main(String[] args)

{

String string1="ace";

String string2= "bdf";

String temp= string1+string2;

String[] listBase =temp.split("");

String[] result = *quickSort*(listBase);

StringBuilder builder = **new** StringBuilder();

**for** (String string : result) {

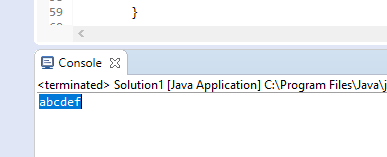
builder.append(string);

}

System.***out***.println( builder );

}

}



Problem2

**package** problem2;

**import** java.util.ArrayList;

**public** **class** Solution2 {

**public** **static** String[] PivotHelper(String[] array, String pivot, Boolean less) {

ArrayList<String> temp = **new** ArrayList<String>();

**if** (array.length <= 1 || array == **null**) {

**return** temp.toArray(**new** String[temp.size()]);

}

**for** (**int** i = 0; i < array.length; i++) {

**if** (less) {

**if** (array[i].hashCode() < pivot.hashCode())

{

temp.add(array[i]);

}

}

**else** {

**if** (array[i].hashCode() > pivot.hashCode())

{

temp.add(array[i]);

}

}

}

**return** temp.toArray(**new** String[temp.size()]);

}

**public** **static** String[] JoinHelper(String[] left, String pivot, String[] right) {

ArrayList<String> temp = **new** ArrayList<String>();

**for** (**int** i = 0; i < left.length; i++)

{

temp.add(left[i]);

}

temp.add(pivot);

**for** (**int** i = 0; i < right.length; i++) {

temp.add(right[i]);

}

**return** temp.toArray(**new** String[temp.size()]);

}

**public** **static** String[] quickSort(String[] array) {

**if** (array.length <= 1) {

**return** array;

}

String pivot = array[0];

String[] left = *quickSort*(*PivotHelper*(array, pivot, **true**));

String[] right = *quickSort*(*PivotHelper*(array, pivot, **false**));

**return** *JoinHelper*(left, pivot, right);

}

**public** **static** **void** main(String[] args)

{

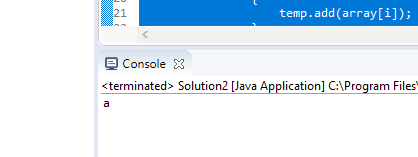
String string1="akel";

String[] result = *quickSort*(string1.split(""));

System.***out***.println( result[0] );

}

}



Problem3

**package** Problem3;

**public** **class** Solution3 {

**public** **static** **int** binarySearch(**int**[] database, **int** index, **int** databaseSize, **int** goal)

{

**if** (databaseSize >= index) {

**int** mid = index + (databaseSize - index) / 2;

**if** (database[mid] == goal) {

**return** mid;

}

**if** (database[mid] > goal) {

**return** *binarySearch*(database, index, mid - 1, goal);

}

**return** *binarySearch*(database, mid + 1, databaseSize, goal);

}

**return** -1;

}

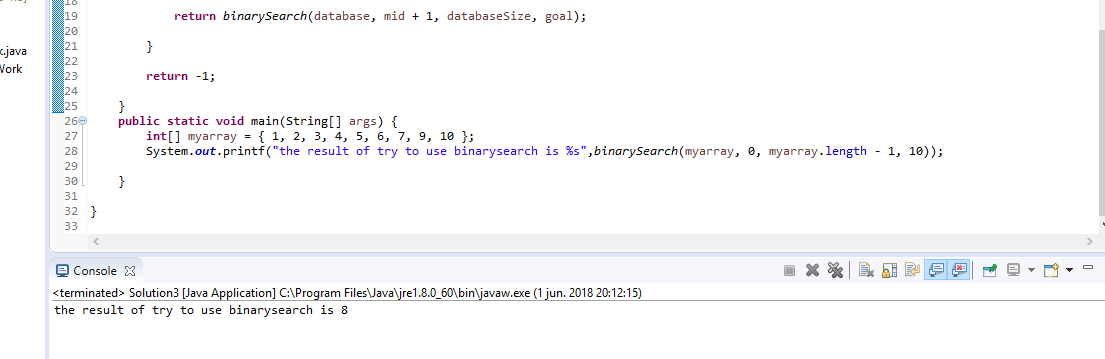
**public** **static** **void** main(String[] args) {

**int**[] myarray = { 1, 2, 3, 4, 5, 6, 7, 9, 10 };

System.***out***.printf("the result of try to use binarysearch is %s",*binarySearch*(myarray, 0, myarray.length - 1, 10));

}

}



Problem4

**package** Problem4;

**public** **class** Solution4 {

**public** **static** Boolean isPalindrome(String string)

{

StringBuilder builder = **new** StringBuilder();

**for** (**int** i = string.length() -1 ; i !=-1 ; i--)

{

builder.append( String.*valueOf*(string.charAt(i)));

}

**return** string.equalsIgnoreCase(builder.toString()) ;

}

**public** **static** **void** main(String[] args)

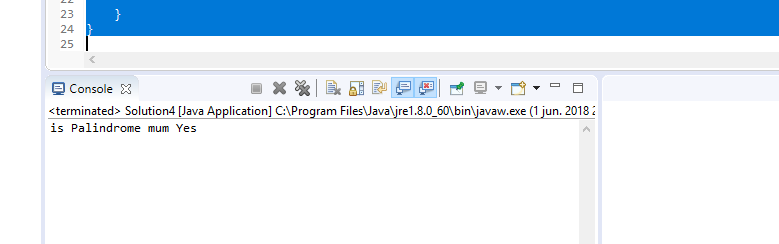
{

String string= "mum";

System.***out***.printf("is Palindrome %s %s",string , ( *isPalindrome*(string) ? "Yes" : "No" ));

}

}



Problem5

import static org.junit.Assert.\*;

import org.junit.Test;

import Problem3.Solution3;

import Problem4.Solution4;

public class UniTestHomeWork

{

@Test

public void goodBinarySearch()

{

int[] database = { 1, 2, 3, 4, 5, 6, 7,8, 9, 10 };

int find = 10 ;

int expected = 9;

assertEquals(expected, Solution3.binarySearch(database, 0, database.length, find));

}

@Test

public void badBinarySearch()

{

int[] database = { 1, 2, 3, 4, 5, 6, 7,8, 9, 10 };

int find = 10 ;

int expected =8;

assertNotEquals(expected, Solution3.binarySearch(database, 0, database.length, find));

}

@Test

public void badProblem4()

{

assertNotEquals(false, Solution4.isPalindrome("Mum"));

}

@Test

public void goodProblem4()

{

assertEquals(true, Solution4.isPalindrome("Mum"));

}

}

