GRADLE USER MANUAL

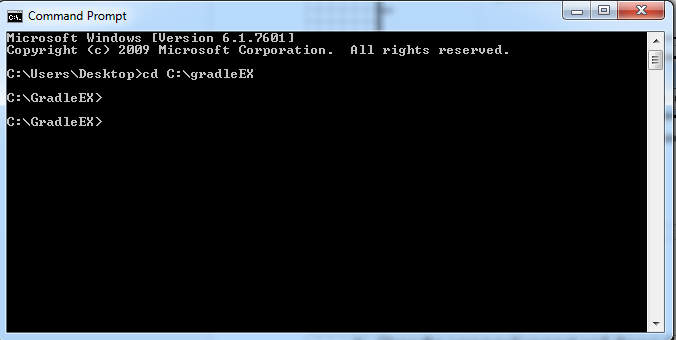
BY: GROUP 2

ICS 312L |PROGRAMMING APPLICATIONS | 4:00-5:30 WS

HOW TO USE THE COMMANDS

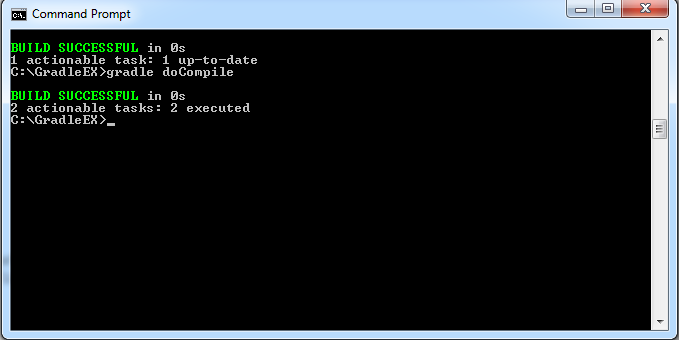
FOR COMPILING THE CODE:

1. Open the command prompt and change the directory to the folder where your java file is located.

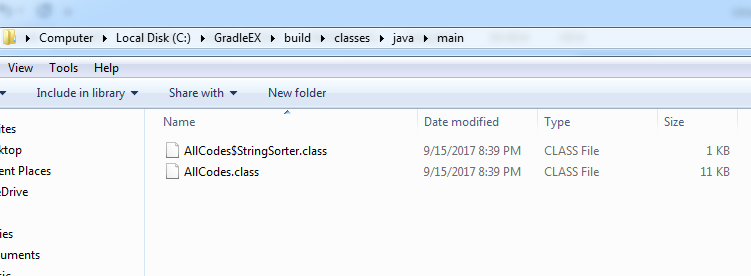
Example: C:\GradleEX

1. Type “gradle doCompile” to compile the java file

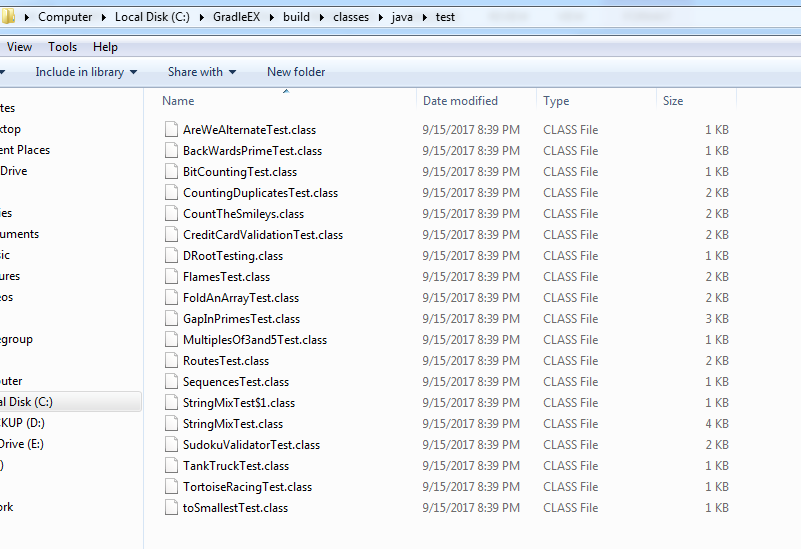
(Note: The command is not case-sensitive. )



\*\* A “build” folder will be created automatically and the “classes” folder will be made as a subfolder and inside it are the classes for the java files. \*\*

 Example:

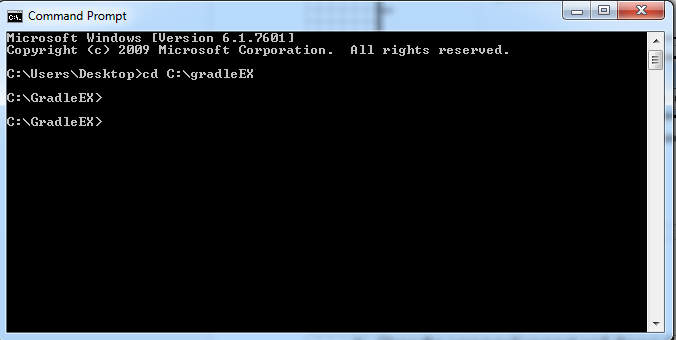




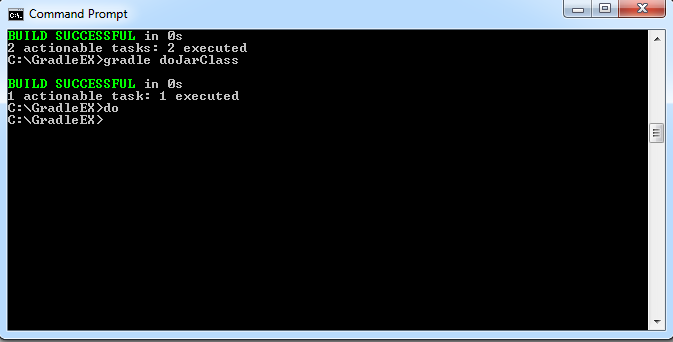


FOR CREATING A JAR CLASS:

1. Open the command prompt and change the directory to the folder where your java file is located.

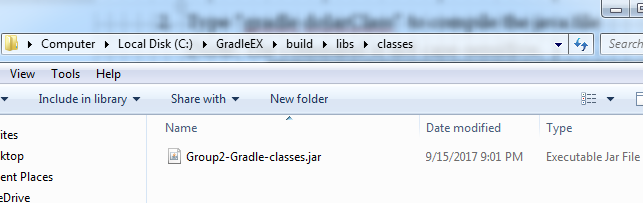
Example: C:\GradleEX

1. Type “gradle doJarClass” to create a jar class

(Note: The command is not case-sensitive. )

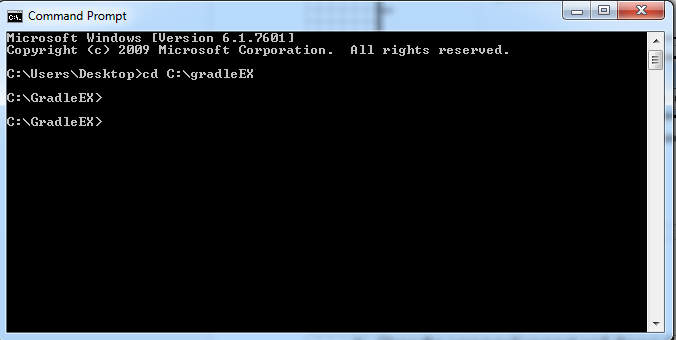
\*\* A “libs” and “classes” subfolder will be created under the build folder. The jar class will be located in the “classes”folder. \*\*

Example:



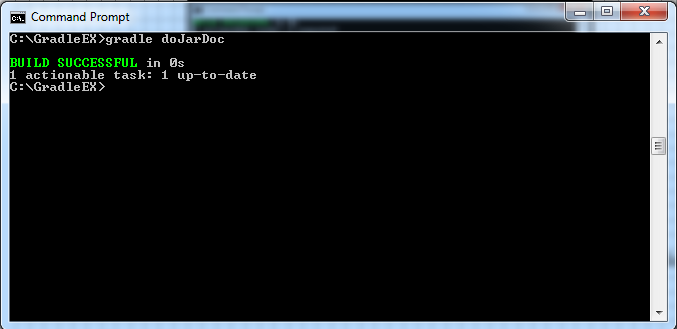
FOR CREATING A JAR DOC:

1. Open the command prompt and change the directory to the folder where your java file is located.

Example: C:\GradleEX

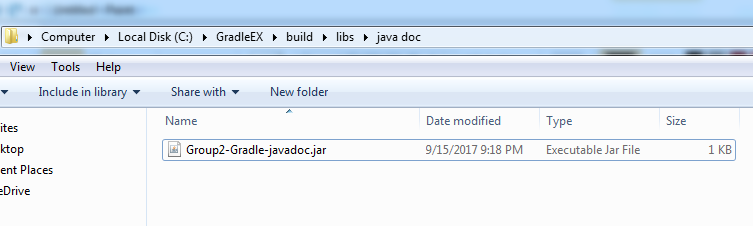
1. Type “gradle doJarDoc” to create a jar doc

(Note: The command is not case-sensitive. )



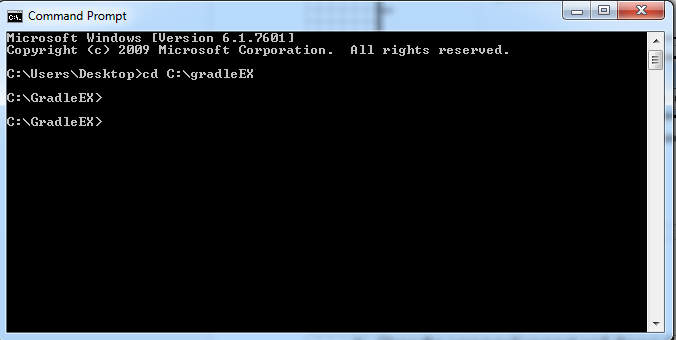
\*\* A “libs” and “java doc” subfolder will be created under the build folder. The jar class will be located in the “java doc”folder. \*\*

Example:

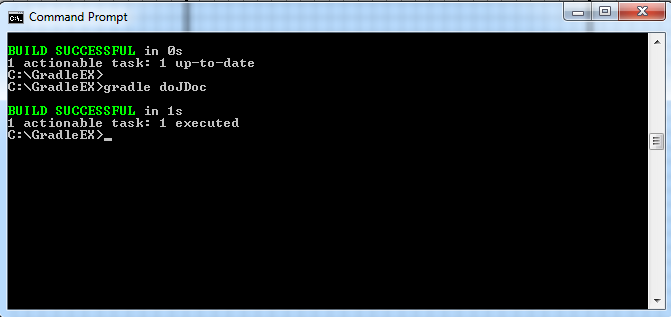


FOR CREATING A JAVA DOC:

1. Open the command prompt and change the directory to the folder where your java file is located.

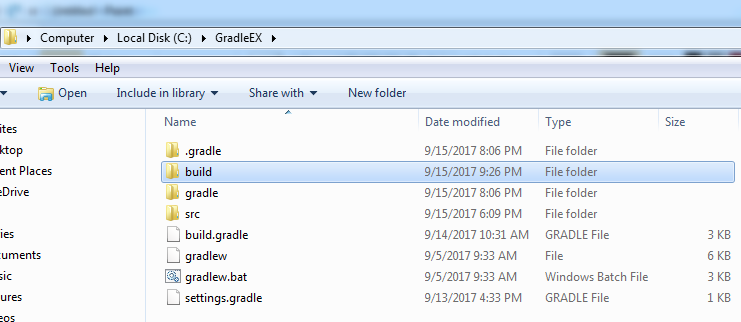
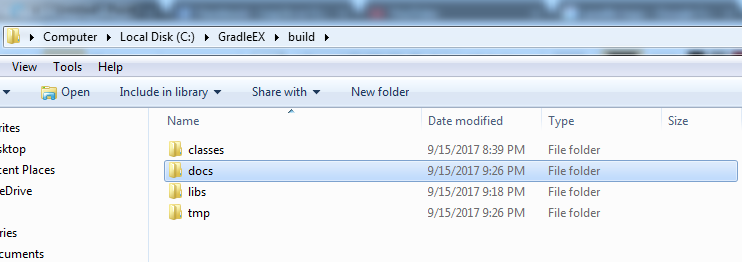
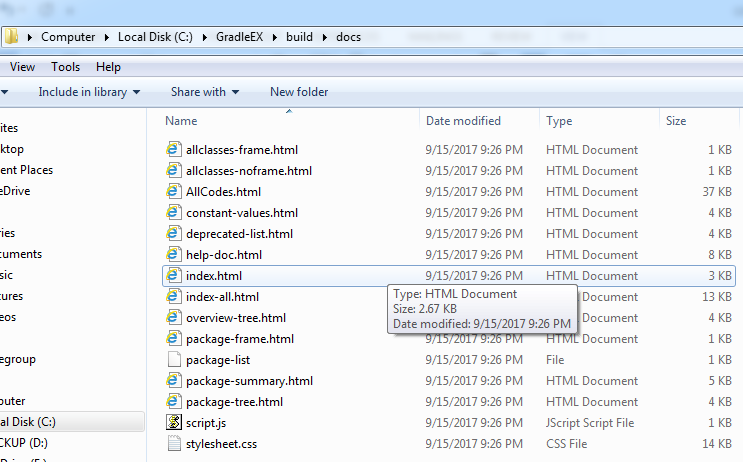
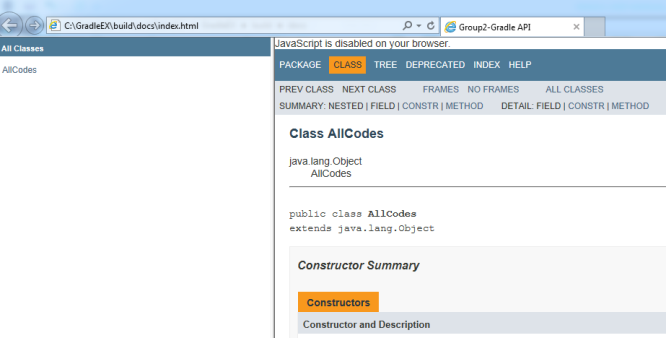
Example: C:\GradleEX

2. Type “gradle doJDoc” to create a jar doc

(Note: The command is not case-sensitive. )

1. Click the “index.html” link located at “\build\docs” to view the document

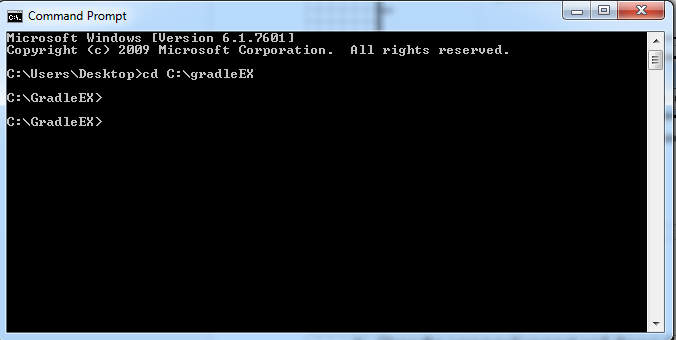
Example: C:\GradleEX\build\docs



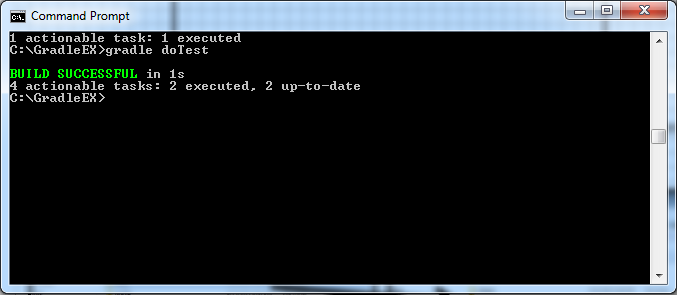


FOR UNIT TESTING:

1. Open the command prompt and change the directory to the folder where your java file is located.

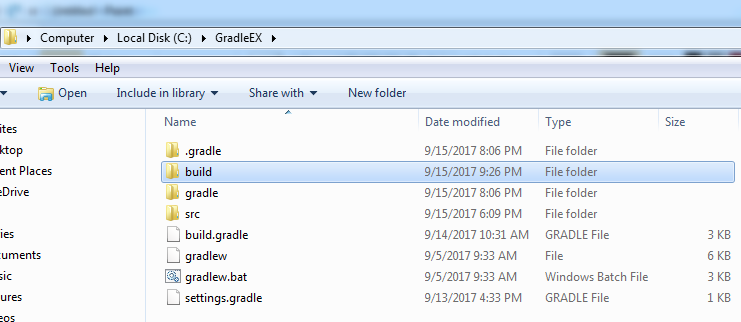
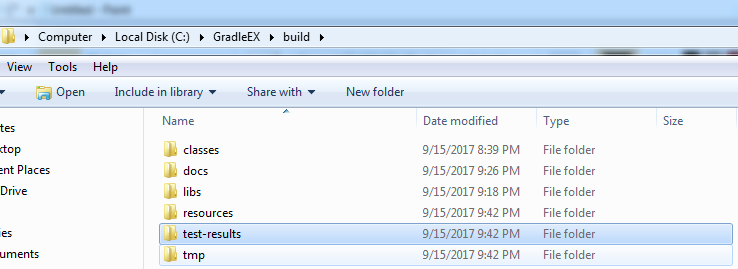
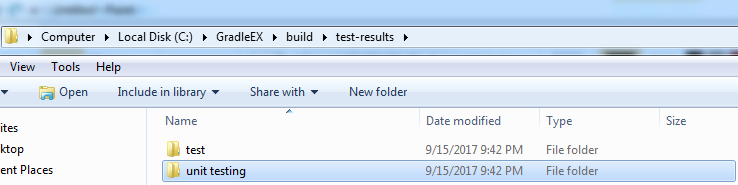
Example: C:\GradleEX

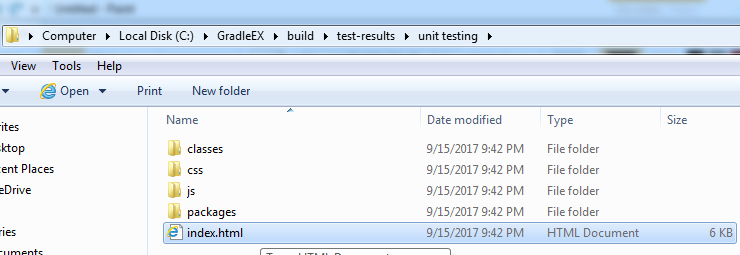
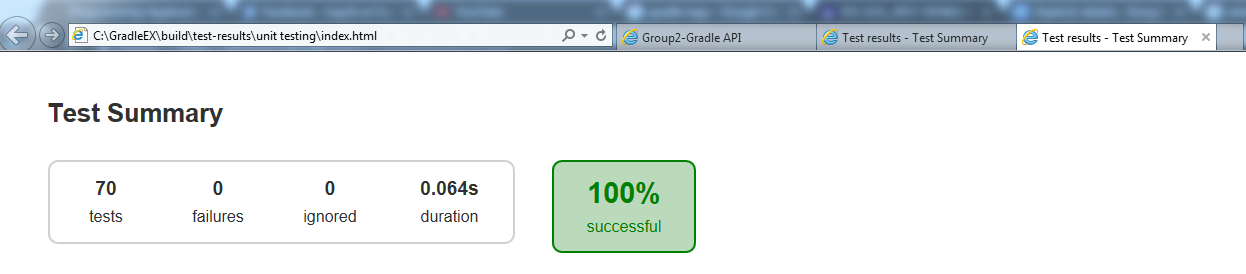
2. Type “gradle doTest” to execute the unit testing

(Note: The command is not case-sensitive. )

1. Click the “index.html” link located at “\build\test –results\unit testing” to view the results of the unit testing.

Example: C:\GradleEX\build\test-results\unit testing

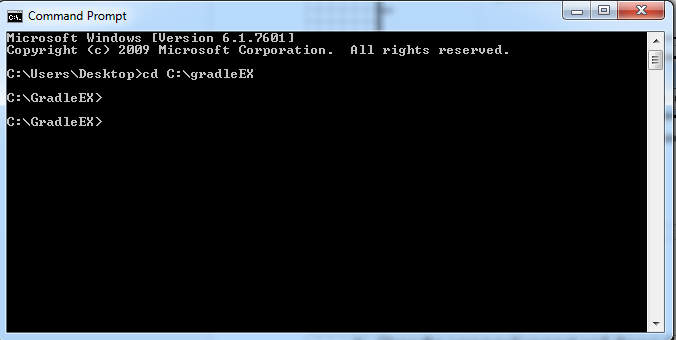


****

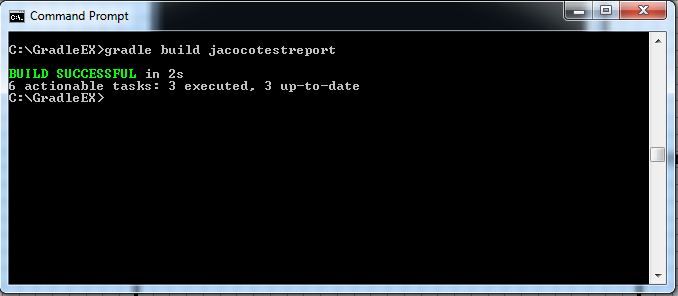


FOR TEST COVERAGE:

1. Open the command prompt and change the directory to the folder where your java file is located.

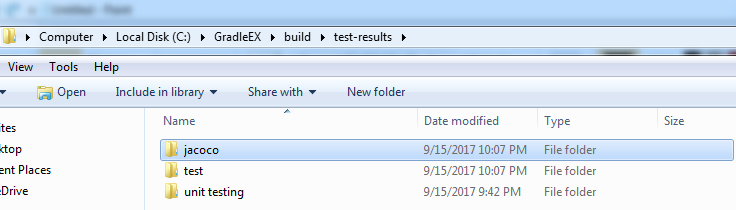
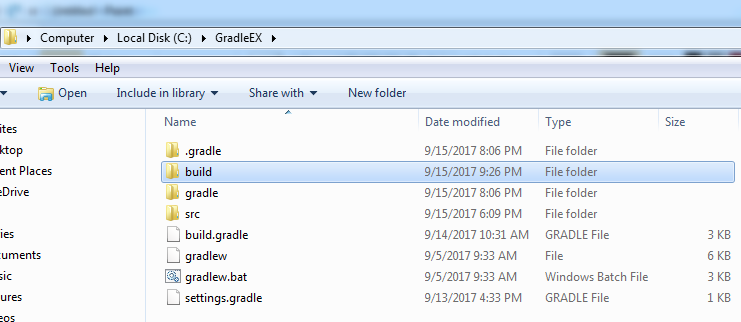
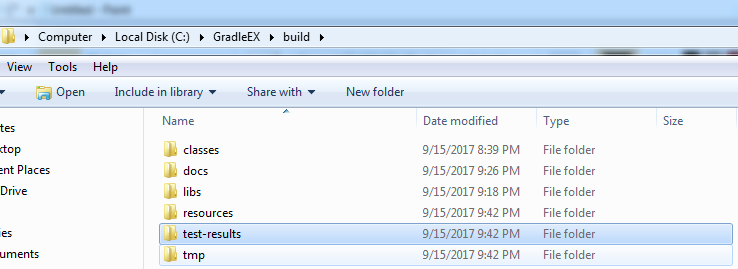
Example: C:\GradleEX

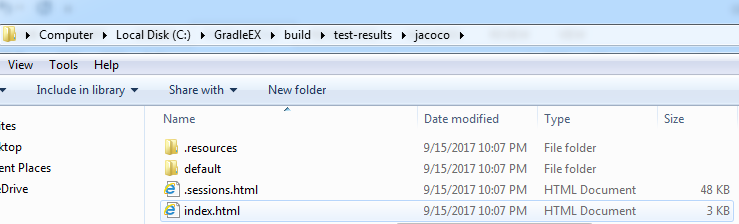
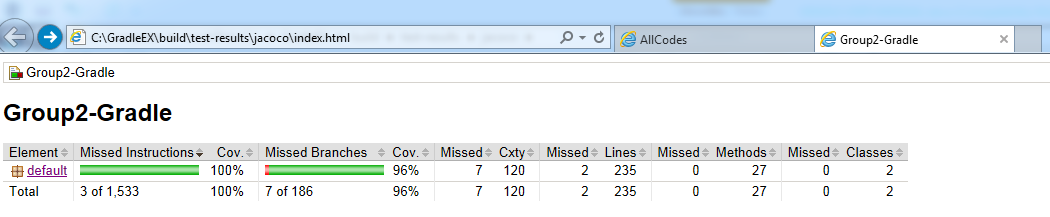
1. Type “gradle build jacocotestreport” to execute java code coverage

(Note: The command is not case-sensitive. )

1. Click the “index.html” link located at “\build\test –result\jacoco” to view the results of the code coverage.

Example: C:\GradleEX\build\test-results\unit testing

****

****



APPENDIX

AllCodes.Java

import java.util.\*;

public class AllCodes {

/\* PROBLEM: TOSMALLEST \*/

/\*\*

\*Algorithm:

\*1.Declare long minimum, that will be used to store the lowest number possible

\*2.Declare int index taht will hold the index of the lowest number in n

\*3.Declare int moveToIndex, that will hold the index where you move the lowest number in n

\*4.Declare String number that will hold the String form of n

\*5.for int i = 0; i is less than length of number; increment i

\*5.1.for int j=0; j is less than length of numberl increment j

\*5.1.1.if i is not equal to j making(number, i, j) is less than minimum

\*5.1.1.1. assign making(number, i, j) to minimum

\*5.1.1.2. assign i to index

\*5.1.1.3. assign j to moveToIndex

\*6. return new long[]{minimum, index, moveToIndex}

\*@param n is the number that you will need to rearrange to have the lowest number by moving only one number

\*@return new long[] that will return the edited number, index of the smallest number and the index where you the smallest number

\*/

public static long[] smallest(long n) {

long minimum = n;

int index = 0;

int moveToIndex = 0;

String number = String.valueOf(n);

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

for (int j=0; j<number.length(); j++) {

if (i!=j && making(number, i, j) < minimum) {

minimum = making(number, i, j);

index = i;

moveToIndex = j;

}

}

}

return new long[]{minimum, index, moveToIndex};

}

/\*\*

\*Algorithm:

\*1. Let sb be a StringBuilder and instantiate it

\*2. Let c be a char that will hold the character at given index of sb

\*3. delete the character of sb at given index

\*4. insert c in sb at given moeToIndex

\*5. Return Long.valueOf(sb.toString()

\*@param number is the number to be edited

\*@param index is the index where the smallest number in number is found

\*@param moveToIndex is the index where you move the smallest number

\*@return value of the string

\*/

public static long making(String number, int index, int moveToIndex) {

StringBuilder sb = new StringBuilder(number);

char c = sb.charAt(index);

sb.deleteCharAt(index);

sb.insert(moveToIndex, c);

return Long.valueOf(sb.toString());

}

/\* PROBLEM: TORTOISE \*/

/\*\*

\*Algorithm for race method

\*1. Declare an int variable for hr, min , sec

\*2.1 If V1 is less than or equal to V2, return null

\*2.2 ELSE get the value of sec by subtracting

\* V2 to V1 (v2-v1) then divide it to the product

\* of 3600 and G (3600\*g) 3600\*g/(v2-v1)

\*2.3 get the value of hr by dividing the value of

\* second to 3600 (sec/3600)

\*2.4 get the value of the new sec by subtracting

\* the product of 3600 \* hr (sec-3600\*hr)

\*2.5 get the value of min by diving sec to 60

\* (sec/60)

\*2.6 get the value of sec by subtracting sec to

\* the product of 60 and min (sec - 60 \* min)

\*2.7 return the new array of hr,min,sec {hr,min,sec }

\*

\*@param v1 is the integer velocity of tortoise A

\*@param v2 is the integer velocity of tortoise B

\*@param g is the integer lead of tortoise A

\*@return int[] of time {hr,min,sec} of how long B will catch A

\*\*/

public static int[] race(int v1, int v2, int g) {

int hr = 0;

int min= 0;

int sec = 0;

if(v1 >= v2){

return null;

}

sec = 3600 \* g / (v2 - v1);

hr = sec / 3600;

sec= sec - 3600 \* hr;

min = sec / 60;

sec = sec - 60 \* min;

System.out.print(new int[]{hr,min,sec});

return new int[]{hr,min,sec};

}

/\* PROBLEM: TANKTRUCK \*/

/\*\*

\* Algorithm:

\* 1. Declare double radius equivalent to the radius,divide the diameter by 2

\* 2. Declare double radiusSquared equals to the value of the radius squared

\* 3. Declare double heightSquared equivalent to the height squared

\* 4. Declare double length equivalent to the maximum volume of the tank divided by area of the cylinder top

\* 5. Declare double equivalent answer equivalent to the computed reamining value

\* 6. Declare int result as answer

\* 7. Return result

\* @param h the height of the tank

\* @param d the diameter of the tank

\* @param vt the maximum volume of the tank

\* @return result the remaining volume left in the tank

\*/

public static int tankVol(int h, int d, int vt) {

double radius=1.0f\*d/2;

double radiusSquared=Math.pow(radius,2);

double heightSquared=Math.pow(h,2);

double length=vt/(Math.PI\*radiusSquared);

double answer=length\*(radiusSquared\*(Math.acos((radius-h)/radius))-(radius-h)\*(Math.sqrt(2\*radius\*h-heightSquared)));

int result=(int)answer;

return result;

}

/\* PROBLEM: SUKOKU VALIDATOR \*/

/\*\*

\*4 kyu Sudoku Solution Validator

\*

\*Algorithm:

\* 1. Construct a loop for row and column of the two dimensional array

\* 2. Declare a boolean variable and set it to true

\* 3. Construct nested loops for the row and column of the two dimensional array

\* 4. If there is an equal value in one row or a value that is equal to 0, set the variable to false

\* 5. Return the variable

\* @param s integer of the 2-D array

\* @return a the validated answer solution

\*/

public static boolean check(int[][] s) {

boolean a = true;

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

for (int j = 0; j < s[i].length; j++) {

for(int k = j+1; k<s[i].length-1; k++) {

if(s[i][k]==s[i][j] || s[i][k]==0) {

a = false;

}

}

}

}

return a;

}

/\* PROBLEM: STRING MIX\*/

/\*\*

\* <p>

\* Receives two string parameters which then puts them into a process that would later show the highest occurrence of each letter the appears

\* more than twice.

\* </p>

\*

\* Algorithm:

\* <ul style="list-style-type:none">

\* <li>1. Store the number of occurrence per character in two different integer array of the two strings.

\* <li>2. Check if the number of occurrence per character is greater than the other.

\* <ul style="list-style-type:none">

\* <li>2.1 Concatenate "1:", "2:" or "=:", depends on which is higher, to a string array.

\* <li>2.2 Keep on concatenating the letter until the number reaches it's occurrences.

\* </ul>

\* <li>3. Append the string array into one string and return then output.

\* </ul>

\*

\* @param s1, String representation of the first sentence.

\* @param s2, String representation of the second sentence.

\* @return String, mixture of s1 and s2 that shows the occurrence of letters in both strings.

\*/

public static String mix(String s1, String s2) {

char[] alphabet = new char[]{'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z'};

char[] a = s1.replaceAll(" ", "").replaceAll("[^a-z]", "").toCharArray();

int[] aa = new int[26];

char[] b = s2.replaceAll(" ", "").replaceAll("[^a-z]", "").toCharArray();

int[] bb = new int[26];

String[] c = new String[26];

String d = "";

AllCodes newInstance = new AllCodes();

Arrays.sort(a);

Arrays.sort(b);

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

aa[a[i] - 'a']++;

}

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

bb[b[i] - 'a']++;

}

for(int i=0; i<26; i++) {

if(aa[i]>bb[i] && aa[i]>1) {

c[i] = "1:";

for(int j=0; j<aa[i]; j++) {

c[i] += alphabet[alphabet[i]-'a'];

}

} else if(aa[i]<bb[i] && bb[i]>1) {

c[i] = "2:";

for(int j=0; j<bb[i]; j++) {

c[i] += alphabet[alphabet[i]-'a'];

}

} else if(aa[i]==bb[i] && aa[i]>1 && bb[i]>1) {

c[i] = "=:";

for(int j=0; j<bb[i]; j++) {

c[i] += alphabet[alphabet[i]-'a'];

}

} else {

c[i]="";

}

}

Arrays.sort(c, newInstance.new StringSorter());

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

if(c[i]=="") {

break;

}

d += c[i]+"/";

}

return d = d=="" ? "" : d.substring(0, d.length()-1);

}//mix

/\*\*

\* A custom comparator meant to satisfy the sorting needs of the program. It first compares the strings own length

\* and then compares them lexicographically.

\*/

class StringSorter implements Comparator<String> {

public int compare(String s1, String s2) {

if (s1.length() < s2.length()) {

return 1;

}

if (s1.length() > s2.length()) {

return -1;

}

return s1.compareTo(s2);

}

}//Comparator

/\* PROBLEM: SEQUENCES AND SERIES \*/

/\*\*

\* Solution for: Sequences and Series by BattleRattle

\*

\* @param n the number to be tested

\* @return long the result of the tested number

\*/

public static long getScore(long n) {

return (((n\*(n+1)\*50)/2));

}

/\* PROBLEM: MULTIPLES OF 3 AND 5 \*/

/\*\*

\*6 kyu Muliples of 3 and 5

\*

\*Algorithm:

\* 1. Declare an integer

\* 2. Construct a loop that ends when the value of i is greater

\* or equal to the number.

\* 3. Check if i%3 is equal to 0 or i%5 is equal to zero

\* 3.1 If true, sum = sum + i

\* 4. Return sum

\* @param number integer to be tested

\* @return sum of the multiples of 3 and 5

\*/

public static int solution(int number) {

int sum = 0;

for(int i = 0; i<number; i++){

if(i%3==0 || i%5==0){

sum += i;

}

}

return sum;

}

/\* PROBLEM: GAPINPRIMES \*/

/\*\*

\* <p>

\* Finds the first pair of prime numbers that corresponds to the given gap, and that has no prime numbers in between.

\* </p>

\*

\* Algorithm:

\* <ul style="list-style-type:none">

\* <li>1. Start at the lower limit to the upper limit and check if it is prime.

\* <ul style="list-style-type:none">

\* <li>1.1 If the number is prime then check if the gap+1 is prime and if there's no prime numbers in between of them.

\* <ul style="list-style-type:none">

\* <li>1.1.1 If there's none then return the current pair.

\* <li>1.1.2 Else, return null.

\* </ul>

\* </ul>

\* </ul>

\*

\* @param gap, integer value of the wanted gap between to primes.

\* @param lLimit, lower limit to which where to start the search.

\* @param uLimit, upper limit to which where to stop the search.

\* @return an array of long that has the value of the pair prime number.

\*/

public static long[] firstGap(int gap, long lLimit, long uLimit) {

for(long i=lLimit; i<=uLimit; i++) {

if(isPrime(i)) {

long j = gap + i;

if (isPrime(j) && !repeatedIsPrime(i + 1, j - 1)) {

return new long[]{i, j};

}

}

}

return null;

}//firstGap

/\*\*

\* Checks if there's a prime number between a lower limit and a higher limit.

\*

\* @param x, lower limit

\* @param y, upper limit

\* @return boolean value of whether there's a prime number between x and y.

\*/

public static boolean repeatedIsPrime(long x, long y) {

for(long i=x; i<=y; i++) {

if(isPrime(i)) {

return true;

}

}

return false;

}//repeatedIsPrime

/\* PROBLEM: FOLDANARRAY \*/

/\*

\*Algorithm

\*1. Declare an empty array dummy.

\*2. While runs isn't equal to 0

\* 2.1 Declare an integer value length equevalent to the legth of the array

\* 2.2 If the given array is divisible by two, equate dummy to the length

\* 2.2.1 Else equate dummy to length adn add 1

\* 2.3 Divde the array into two, equate dummy to the result of adding the values of the two halves

\* 2.4 If length of the dummy array is equal to halve the length of the array

\* 2.4.1 Decrement runs by 1

\* 2.4.2 Equate array to dummy

\* 2.4.3 Return array

\*@param

\*\*/

public static int[] foldArray(int[] array, int runs){

int[] dummy=null;

while(!(runs==0)){

int length=array.length;

if(array.length%2==0){

dummy=new int[length/2];

}else{

dummy=new int[length/2+1];

}

for(int first=0,last =array.length-1; first<last&&!(first==last); first++,last--){

dummy[first]=array[first]+array[last];

}

if(array.length%2==1)dummy[dummy.length-1]=array[length/2];

runs--;

array=dummy;

}

return array;

}

/\* PROBLEM: FLAMES \*/

/\*\* Algorithm:

\* 1. Declare initial variables that will hold for the values of the names' combination,

\* its result, and count.

\* 2. For every input, it should be in lowercase form

\* 3. Declare a variable copy to hold the original letters of female input to use later

\* 4. Use a for loop to check the length of the male string

\* 4.1 Replace all the common letters of female string and male string to ""

\* 5. Use a for loop to check the length of the female string (copy)

\* 5.1 Replace all the common letters of male string and the original letters of female string (copy) to ""

\* 6. Add the combined result of the previous loop and store it to the variable combined

\* 7. Count the length of the result and store it to the variable count

\* 8. Instead of cycling through all the letters, use modulo to determine the remainder and use it to count.

\* 9. If result is greater than 6 or result is equal to 0, equate result to 6

\* 10. Checks the result in the following array of strings

\* @param male the string to be inputed for the male

\* @param female the string to be inputed for the female

\* @return the result for the flames

\*\*/

public static String showRelationship(String male, String female) {

String combined = "";

int result = 0;

int count = 0;

female = female.toLowerCase();

male = male.toLowerCase();

String copy = female;

for(int x = 0; x < male.length(); x++){

female = female.replaceAll(String.valueOf(male.charAt(x)), "");

}

for(int x = 0; x < copy.length(); x++){

male = male.replaceAll(String.valueOf(copy.charAt(x)), "");

}

combined = male + female;

count = combined.length();

result = count%6;

if(result > 6 || result == 0){

result = 6;

}

String[] flames = {"Friendship", "Love", "Affection", "Marriage", "Enemies", "Siblings"};

return flames[result-1];

}

/\* PROBLEM: DROOT \*/

/\*\*

\*Algorithm for digital\_root method

\*1. while the input integer is less than 9,

\* do:

\* get the value of n by adding the

\* dividend of n and 10 (n/10) and

\* the remainder of n modulo 10 ( n/10 + n % 10)

\*2. if N is less than 9 , return the value of n

\*@param n is the integer to be inputed

\*@return n is the recursive sum of all digits

\*\*/

public static int digital\_root(int n) {

while(n > 9){

n = n/10 + n % 10;

}

return(n);

}

/\* PROBLEM: CREDITCARDVALIDATION \*/

/\*\*

\* <p>

\* Puts the given string through a number of processes which would then result into a boolean value

\* that tells if it is a valid credit card number or not.

\* </p>

\*

\* Algorithm:

\* <ul style="list-style-type:none">

\* <li>1. Get the numeric values of each character in cNum and store it in nNum.

\* <li>2. Starting from the furthest right and going left by two's check if the doubled value of

\* nNum[j] is greater than 9.

\* <ul style="list-style-type:none">

\* <li>2.1. True, nNum[j] will now be equal to nNum[j]\*2-9.

\* <li>2.2. False, nNum[j] will now be equal to nNum[j]\*2.

\* </ul>

\* <li>3. Get the summation of all indexes of the array nNum and store it ti tSum.

\* <li>4. Check if tSum%10==0.

\* <ul style="list-style-type:none">

\* <li>4.1. True, return true.

\* <li>4.2. False, return false.

\* </ul>

\* </ul>

\* @param cNum, String representation of the credit card number to be processed.

\* @return Boolean value of whether the input cNum was valid or not.

\*/

public static Boolean validate(String cNum) {

int tSum = 0;

int[] nNum = new int[cNum.length()];

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

nNum[i] = Character.getNumericValue(cNum.charAt(i));

}

for(int j=nNum.length-2; j>=0; j-=2) {

nNum[j] = nNum[j]\*2>9 ? nNum[j]\*2-9 : nNum[j]\*2;

}

for(int k=0; k<nNum.length; k++) {

tSum += nNum[k];

}

return ((tSum%10)==0);

}//validate

/\* PROBLEM: COUNTING DUPLICATES\*/

public static int duplicateCount(String text) {

/\*\*

\* 1. Variable originaltext copies the orginaltext from the parameter

\* 2. Text is then converted to its lower case

\* 3. Create and initialize a stringbuilder to copy the text

\* 4. Create and initialize a counter for a character

\* 5. Create a loop that will check for the length of the stringbuilder

\* 6. Create and initialize a char variable to check the first character of the string.

\* 7. Check if the first character is not out of range, then delete and increment the count variable

\* 8. If a character occurs more than once increment the count variable else display none is repeated.

\*/

String originaltext=text;

text=text.toLowerCase();

StringBuilder sb = new StringBuilder(text);

int count2=0;

while(sb.length() != 0)

{

int count = 0;

char test = sb.charAt(0);

while(sb.indexOf(test+"") != -1)

{

sb.deleteCharAt(sb.indexOf(test+""));

count++;

}

if(count>1)

{

count2++;

//System.out.println(originaltext+"-> "+count2+" "+test+" occurs "+count+" times");

}

if(count==0)

{

//System.out.println("no characters repeats more than once");

}

}

return count2;

}

/\* PROBLEM: COUNT THE SMILEYS \*/

/\*

\*Solution for Count The Smileys

\*1. Check each index of the given array

\*2. If the length of the string is 2

\* 2.1 Check if the index contains the ff: ":", ";" and ")","D"

\* 2.2 If yes, count + 1

\*3. If the length of the string is 3

\* 3.1 Check each of the index if it contains the ff: ":" ";" "-" "~" ")" "D"

\* 3.2 If yes, count + 1

\*

\*@param list of the array that will contain the strings to be counted

\*@return number of the smileys given in the array \*/

public static int countSmileys(List<String> arr) {

int count = 0;

for(int i=0; i< arr.size(); i++){

if(arr.get(i).length() == 2){

if((arr.get(i).indexOf(':') == 0 || arr.get(i).indexOf(';') == 0) && (arr.get(i).indexOf(')') == 1 || arr.get(i).indexOf('D') == 1)){

count++;

}

} else if(arr.get(i).length() == 3){

if((arr.get(i).indexOf(':') == 0 || arr.get(i).indexOf(';') == 0) && (arr.get(i).indexOf('-') == 1 || arr.get(i).indexOf('~') == 1) && (arr.get(i).indexOf(')') == 2 || arr.get(i).indexOf('D') == 2)){

count++;

}

}

}

return count;

}

/\* PROBLEM: BITCOUNTING \*/

/\*\*

\* This method converts the inter into binary

\* counts all of the 1's in the binary

\* prints the number of 1's

\* @param n This is the number to be tested

\* @return int This returns the number of 1's in a binary

\*/

public static int countBits(int n){

return Integer.bitCount(n);

}

/\* PROBLEM: BACKWARDSPRIME \*/

/\*\*

\*Algorithm:

\*1.Let result be a String that will hold the result of the method

\*2.Let reversedPrime be a long that will hold the reverse of the prime number

\*3.long i = start; i less than or equal end; i++

\*3.1. if i is prime

\*3.2. assign the reverse number of i to reversedPrime

\*3.3. if reversedPrime is prime and i is not equal to reversedPrime

\*3.3.1. result+=i+""

\*4. return result.trim()

\* @param start is the start of the number you will need to find a prime number

\* @param end is the upper limit of the number you will need to find a prime number

\* @return String result that is the concatitaion of all the primes from start to end

\*/

public static String backwardsPrime(long start, long end) {

String result = "";

long reversedPrime= 0;

for(long i = start; i <= end; i++){

if(isPrime(i)){

reversedPrime = reverseNumber(i);

if(isPrime(reversedPrime) && !Long.toString(i).equals(new StringBuilder(Long.toString(reversedPrime)).toString())){

result += i+" ";

}

}

}

return result.trim();

}

/\*\*

\*Algorithm:

\*1. if number modulo 2 is equal to 0

\*1.1. return false

\* else

\*2.for(int i = 3; i\*i less than or equal number; i+=2)

\*2.1.if number modulo i is equal to 0

\*2.1.1. return false

\*3.return true

\* @param number is the number that will be tested if it is prime

\* @return boolean will return true if number is prime else false

\*/

public static boolean isPrime(long number){

if(number%2 == 0){

return false;

}else{

for(int i = 3; i\*i <= number; i+=2){

if(number%i == 0){

return false;

}

}

}

return true;

}

/\*\*

\*Algorithm:

\*1.Let reversedNum be a long that will hold the reverse of num

\*2.while num is not equal to 0

\*2.1.Let remainder be a long that will hold the value of num modulo 10

\*2.2.Assign reversedNum\*10+remainder to reversedNum

\*2.2.Assign num/10 to num

\*3.return reversedNum

\*@param num is the number to be reversed

\*@return long reverseNumber is the reversed number of num

\*/

public static long reverseNumber(long num){

long reversedNum = 0;

while(num != 0){

long remainder = num%10;

reversedNum = reversedNum\*10+remainder;

num = num/10;

}

return reversedNum;

}

/\* PROBLEM: ROUTES \*/

/\*\*

\* Solution for: 'Follow that spy' by adromil

\* Algorithm:

\* 1. Receive parameter value of routes

\* 2. Declare String[] variable temp1, int variable counter, String variable temp3, boolean

\* variable spyFollowed, ArrayList(String[]) variable oldRoute, ArrayList(String) variable

\* newRoute

\* 3. Copy the contents of routes to oldRoute

\* 4. Add the values of first array of oldRoute to newRoute, then removes said array

\* 5. While spyFollowed is false:

\* 5.1 Copy first array of oldRoute to temp1

\* 5.2 Compare temp1[1] String value to the first String value of newRoute

\* 5.3 If String values are equal:

\* 5.3.1 Insert temp1[0] value at index 0 of newRoute, remove array from oldROute, and increment counter by 1

\* 5.3.2 Check if oldRoute is empty. If empty, set spyFollowed to true

\* 5.3 If String values are not equal, increment counter by 1 and skip to next array

\* 5.4 If counter is equal to the size of oldRoute (no match), proceed to switchRoute method

\* 6. Add all values of newRoute to temp3

\* 7. Return temp3

\*

\* @param routes the array of arrays that will contain the strings to be sorted

\* @return String the sorted string

\*/

public static String findRoutes(String[][] routes) {

String[] temp1; int counter=0; String temp3="";

boolean spyFollowed = false;

ArrayList<String[]> oldRoute = new ArrayList<String[]>();

ArrayList<String> newRoute = new ArrayList<String>(); // empty

for (String[] temp2 : routes) {

oldRoute.add(temp2);

}

temp1 = oldRoute.get(0);

newRoute.add(temp1[0]);

newRoute.add(temp1[1]);

oldRoute.remove(0);

try {

if (oldRoute.isEmpty()) { spyFollowed = true; }

while (!spyFollowed) {

for (int a=0; a<oldRoute.size(); a++) {

temp1 = oldRoute.get(a);

if (temp1[1].equalsIgnoreCase(newRoute.get(0))) {

newRoute.add(0, temp1[0]);

oldRoute.remove(a); counter=0;

if (oldRoute.isEmpty()) { // oldRoute empty

spyFollowed = true;

}

break;

} else if (counter == oldRoute.size()) { // no match, oldRoute !empty

newRoute = switchRoute(newRoute, oldRoute);

spyFollowed = true;

break;

} else {

counter++;

}

}

}

temp3 = newRoute.get(0);

for (int c=1; c<newRoute.size(); c++) {

temp3 += ", " + newRoute.get(c);

}

} catch (Exception e1) {

e1.printStackTrace();

}

return temp3;

}

/\*\*

\* Algorithm:

\* 1. Receives parameter values of newRoute and oldRoute

\* 2. Declare String[] variable temp4, boolean variable switchDone

\* 3. While switchDone is false:

\* 3.1 Copy first array of oldRoute to temp4

\* 3.2 Compare temp4[0] to the last String value of newRoute

\* 3.3 If String values are equal:

\* 4.3.1 Insert temp4[1] value at the end of newRoute and remove array from oldRoute

\* 4.3.2 Check if oldRoute is empty. If empty, set switchDone to true

\* 3.4 If String values are not equal, skip to next array

\* 4. Return newRoute

\*

\* @param newRoute the ArrayList(String) for sorted elements

\* @param oldRoute the ArrayList(String[]) for unsorted elements

\* @return ArrayList(String) the sorted ArrayList(String)

\*/

public static ArrayList<String> switchRoute(ArrayList<String> newRoute, ArrayList<String[]> oldRoute) {

String[] temp4; boolean switchDone = false;

while (!switchDone) {

for (int b=0; b<oldRoute.size(); b++) {

temp4 = oldRoute.get(b);

if (temp4[0].equalsIgnoreCase(newRoute.get(newRoute.size()-1))) {

newRoute.add(temp4[1]);

oldRoute.remove(b);

if (oldRoute.isEmpty()) {

switchDone = true;

}

break;

}

}

}

return newRoute;

}

/\* PROBLEM: AREWEALTERNATE? \*/

/\*

\*Algorithm:

\*<ul>

\*<li>1. Check the first character if its a vowel

\*<li>2. Using a for loop check if every character is a vowel

\*</ul>

\*@param word is the string to be checked

\*@return boolean will tell if its alternate

\*\*/

public static boolean isAlt(String word)

{

boolean b = isVowel(word.charAt(0));

for (int i = 1; i < word.length(); i++)

{

b = !b;

if (b != isVowel(word.charAt(i)))

return false;

}

return true;

}

static boolean isVowel(char ch)

{

return "AEIOUaeiou".indexOf(ch) != -1;

}

}