public class Homework {

1)

public static Integer nFibonacci( Integer n ){

Integer tempnumber = 0;

Integer currentnumber = 0;

Integer pastnumber = 0;

If (n > 0) pastnumber = 1;

do {

tempnumber = currentnumber;

currentnumber += pastnumber;

pastnumber = tempnumber;

n--;

} while (n > 0);

return currentnumber;

}

2)

public static Integer[] sortArray( Integer[] intList ) {

Integer tempInd;

for (Integer i = 0; i < intList.size() - 1; i++) {

for (Integer j = i + 1; j < intList.size(); j++) {

if (intList[i] > intList[j]) {

tempInd = intList[i];

intList[i] = intList[j];

intList[j] = tempInd;

}

}

}

return intList;

}

3)

public static Integer nFactorial( Integer n) {

Integer factorial = 0;

if (n == 0) factorial = 1;

do {

factorial \*= n;

n--;

} while (n >= 0);

return factorial;

}

4)

public static Integer[] rotateLeftNTimes( Integer[] numberArray, Integer n) {

Integer firstInd;

for (Integer i = 0; i < n; i++) {

for (Integer j = 0; j < numberArray.size(); j++) {

if (j == 0) {

firstInd = numberArray[j];

numberArray[j] = numberArray[j + 1];

} else if (j == numberArray.size() - 1) {

numberArray[j] = firstInd;

} else {

numberArray[j] = numberArray[j + 1];

}

}

}

return numberArray;

}

5)

public static Boolean bracketsAreBalanced( String s ) {

String [] brackets = s.split('');

if (Math.mod(brackets.size(), 2) != 0) return false;

for (Integer i = 0; i < brackets.size(); i++) {

for (Integer j = 0; j < brackets.size(); j++) {

switch on brackets[j] {

when ')' {

if (brackets[j-1] == '(') {

brackets.remove(j);

brackets.remove(j-1);

} else {

return false;

}

} when '}' {

if (brackets[j - 1] == '{') {

brackets.remove(j);

brackets.remove(j-1);

} else {

return false;

}

} when ']' {

if (brackets[j - 1] == '[') {

brackets.remove(j);

brackets.remove(j-1);

} else {

return false;

}

}

}

}

}

return true;

}

6)

public static void updateAccountSize( ) {

List <Account> accountList = [select size\_\_c, numberofemployees

from account  
 where numberofemployees != null];

List <Account> accountToUpdate = new List <Account>();

for (Account a : accountList) {

if (a.numberofemployees < 1001) a.size\_\_c = 'small';

else if (a.numberofemployees > 1000 && a.numberofemployees < 10001) a.size\_\_c = 'medium';

else a.size\_\_c = 'large';

accountToUpdate.add(a);

}

update accountToUpdate;

}

7)

public static void updateCALeads( ) {

List <Lead> leadList = [select status, description

from Lead   
 where state = 'CA'];

List <Lead> leadToUpdate = new List <Lead>();

for (Lead l : leadList)

{

l.status = 'Closed - Not Converted';

l.description = 'We are no longer doing business in California.';

leadToUpdate.add(l);

}

update leadToUpdate;

}

8)

public static void closePastDueOpportunities( ) {

List<Opportunity> opportunityList = [select stagename

from opportunity   
 where closedate < Today   
 and stagename != 'Closed Won'];

List<Opportunity> opportunityToUpdate = new List<Opportunity>();

for (Opportunity o : opportunityList) {

o.stagename = 'Closed Lost';

opportunityToUpdate.add(o);

}

update opportunityToUpdate;

}

}