Quiz

1. Find the syntax or logical errors in the following statements:

a)
```
int x = Integer.parseInt(input);
if (x != null) y += x;
```
x est une variable de type primitive (int). Elle ne peut pas être testée avec null.

b)
```
String language = "German";
if (country.equals("Swiss")) {
  if (city.equals("Geneva")) language = "French";
} else if (country.equals("Italy"))
  language = "Italian";
```
"else if" correspond à "if (city.equals("Geneva"), mais devrait correspondre à "if (country.equals("Swiss")." Pour corriger, ajouter les parenthèses {…}

c)
```
if (!input.equals("Y") || !input.equals("y"))
  System.out.println("Refused");
else
  System.out.println("Accepted");
```
Correction:
if (!input.equals("Y") && !input.equals("y"))
OR: if(!input.equalsIgnoreCase("y"))…

d)
```
int x = in.nextInt();
if (in.hasNextInt())
  sum += x;
else
  System.out.println("Input error");
```
Il faut tester l'input AVANT de le lire!

Correction:
```
if (in.hasNextInt()) {
int x = in.nextInt();
sum += x;
} else System.out.println("Input Error");
```

2. Simplify the following Boolean expressions, using the de Morgan's law where appropriate:

a) `!(x > 0 && y > 0)` `x <= 0 || y <= 0`

b) `if (n == 0) b = false; else b = true;` `b = n !=0`

c) `!(x != 0 || y != 0)` `x == 0 && y == 0`

d)
```
!(country.equals("CH") && !city.equals("Bern") &&
                          !city.equals("Biel/Bienne"))
```
`!country.equals("CH") || country.equals("Bern") || city.equals("Biel/Bienne")`

e)
```
b = false; if (n > 1) if (n < 2) b = true;
```
`b = n > 1 && n < 2`

f)
```
if (n < 1) b = true; else b = n > 2;
```
`b = n < 1 || n > 2`

g)
```
!(x % 4 != 0 || !(x % 100 == 0 && x % 400 == 0))
```
`x % 4 == 0 && (x % 100 == 0 || x % 400 == 0)`
`OR x % 400 == 0`

3. How many iterations do the following loops carry out?

e) `for (int i = -10; i <= 10; i++)...` 21

f) `for (int i = -10; i <= 10; i +=2)...` 11

g)
```
for (int i = -10; i <= 10; i +=3)...
```

h)
```
for (int i = Integer.MAX_VALUE - 10;
     i < Integer.MAX_VALUE; i++)...
```
10

i)
```
for (int i = Integer.MAX_VALUE - 10;
     i <= Integer.MAX_VALUE; i++)...
```
endless loop

4. Rewrite the following `do` loop into a `while` loop

```
int n = in.nextInt();
double x = 0;
double s;
do {
   s = 1.0 / (n * n);
   x += s;
   n++;
} while (s > 0.01);
```

(This is easy!)

```
int n = in.nextInt();
double x = 0;
double s = 1; // Trick to make sure the loop passes the first time
while (s > 0.01) {
s = 1.0 / ( n* n);
…
}
```

5. Consider the following array:

```
int[] a = {1, 2, 3, 4, 5, 4, 3, 2, 1, 0};
```

What are the value of `total` after the following loops complete?

a)
```
int total = 0;
for (int i = a.length - 1; i >= 0; i -= 2) total += a[i];
```
total = 12;

b)
```
int total = 0;
for (int i = 0; i < 10; i++) total = a[i] - total;
```
total = -1;

c)
```
int total = 0;
for (int i = 2; i <= 10; i++) total += a[i];
```
ArrayOutOfBoundException: 10

What are the contents of the array after the following loops complete?

a) `for (int i = a.length - 1; i > 0; i--) a[i] = a[i-1];` [1,1,2,3,4,5,4,3,2,1]

b) `for (int i = 1; i < 10; i++) a[i] += a[i-1];` [1,3,6,10,15,19,22,24,25,25]

c) `for (int i = 1; i < a.length/2; i++) a[i] = a[a.length - i]`

d) `for (int i = 1, j = a.length - 1; i < j; i++, j--) a[i] = a[j];`

[1,0,1,2,3,4,3,2,1,0]

[1,0,1,2,3,4,3,2,1,0]

6. Rewrite the following loops, using the "for each" construct.

```
int[][][] a3D = {{{1,2},{3,4},{5,6}},{{7,8},{9,10},{11,12}}};
for (int i = 0; i < a3D.length; i++)
  for (int j = 0; j< a3D[i].length; j++)
    for (int k = 0; k < a3D[i][j].length; k++)
      System.out.println(a3D[i][j][k]);
```

```
for (int[][] x : a3D)
 for (int[] y : x)
  for (int z : y)
   System.out.println(z);
```

7. The following algorithm locates and prints the first element of an array list that is larger than 100.

```
while (pos < values.size() && values.get(pos) <= 100) pos++;
System.out.println(pos < values.size() ?
            ("Value " + values.get(pos) + " found at pos: " + pos) :
             "Value not found");
```

What is wrong with using this loop instead?

```
int k = -1;
for (int pos = 0; pos < values.size(); pos++) {
  if (values.get(pos) > 100) k = pos;
}
System.out.println(k >= 0 ?
            ("Value " + values.get(k) + " found at pos: " + k) :
            "Value not found");
```

1) La boucle 'for' ne stoppe pas lorsque l'élèment > 100 est trouvée.
2) Si l'array contient plusieurs valeurs > 100, la boucle 'for' retourne le dernier et non pas le premier.

8. What is wrong with the following method that aims to fill an array with random numbers?

```
public void fillRandomArray(int[] values, int n) {
  Random generator = new Random();
  int[] numbers = new int[values.length];
  for (int x : numbers)
    x = generator.nextInt(n);
  values = numbers;
}
```

1) La valeur 'x' n'est jamais mémorisée dans l'array numbers.
2) En fin de méthode, values référence numbers qui est une variable locale, donc perdue lorsqu'on quitte la méthode.

9. True or false?

a) All elements of an array are of the same type. true

b) An array index must be an integer. true, must be byte, short, char or int. (not long!)

c) Parallel arrays must have equal lengths. true

d) Two parallel arrays can be replaced by a two-dimensional array. false

e) Elements of different columns in a two-dimensional array can have different types. false

10. How do you perform the following tasks with array lists?

a) Copy one array list to another

b) Fill an array list with the value *n*, overwriting all elements in it.

c) Remove the first half of the list elements.

d) Remove all list elements.

11. Write a method that generates a random permutation of the numbers 1 to 10 (assume that a Random object has been instantiated).

```
10.
a) ArrayList<Integer> dataCopy = new ArrayList<>();
for (int x : dataCopy) dataCopy.add(x);
OR:
ArrayList<Integer> dataCopy = new ArrayList<>(dat);
OR:  'clone()'

b) for (int i = 0; i < data.size(); i++) data.set(i,n);

OR:
   Collections.fill(data, n);

c) for (int i = 0; i < size / 2; i++) data.remove(0);

d) data.clear();
```