COMP108 Data Structures and Algorithms
Lab Exercises (Week 4)
(Suggested solution and Feedback)

w/c 1st March 2021

1. Make sure you have the habit of entering your name and ID onto the beginning of programs.
2. **Task 1:**

   - The task involves checking for each entry `array1[i]` whether it does not exist in `array2[ ]`. So this is similar to sequential search in the lecture notes but with `key` replaced by `array1[i]`. We adapt as follows.

```
j = 0;
flag = false;
while (j < size2 && !flag) {
   if (array1[i] == array2[j])
      flag = true;
   j++;
}
if (!flag)
   System.out.print(array1[i] + " ");
```

   - One very important point here is that we can only conclude that `array1[i]` is not in `array2[ ]` if it does not equal to **ALL** entries. We cannot conclude simply by seeing that it doesn't equal to ONE entry. In other words, the following is WRONG:

```
// This is WRONG
j = 0;
flag = false;
while (j < size2 && !flag) {
   if (array1[i] == array2[j]) {
      flag = true;
      System.out.print("exists");
   } else
      System.out.print("not exists");
   j++;
}
```

   - The above loop deals with one entry in `array1[ ]` forming the inner loop when we handle each entry. In other words, we expect the following structure of the outer loop

```
i = 0;
while (i < size1) {
   // fill in the inner loop here
   i++;
}
```

- Combining, we have the following:

```
static void notExists(int array1[], int size1, int array2[], int size2) {
   int i, j;
   boolean flag;

   i = 0;
   while (i < size1) {
      j = 0; // this has to be inside the outer loop
      flag = false; // this has to be inside the outer loop
      while (j < size2 && !flag) {
         if (array1[i] == array2[j])
            flag = true;
         j++;
      }
      // This if-statement has to be outside the inner loop
      if (!flag)
         System.out.print(array1[i] + " ");
      i++;
   }
   System.out.println();
}
```

Feedback: As mentioned above, the most important point is how to determine "not exist" condition. Furthermore, it is also important that the statement resetting `flag` *to* `false` *has to be* ***inside the outer loop*** *instead of outside.*

3. **Task 2:** In this task, the role of `array1[ ]` and `array2[ ]` has swapped. This time we want to check for each entry `array2[i]` how many times it appear in `array1[ ]`. So this time the inner loop will go through `array1[ ]` one by one while the outer loop goes through `array2[ ]` one by one.

```
static void count(int array1[], int size1, int array2[], int size2) {
   int i, j, count;

   i = 0;
   while (i < size2) {
      j = 0; // this has to be inside the outer loop
      count = 0; // this has to be inside the outer loop
      while (j < size1) {
         if (array1[j] == array2[i])
            count++;
         j++;
      }
      System.out.print(count + " ");
      i++;
   }
   System.out.println();
}
```

Feedback: Again it is important to reset `count` *to 0 inside the outer loop.*

4. **Puzzle:**

(a) Divide the 9 eggs into 3 groups of 3, and name them group A, B, C.

- Take groups A and B to weigh.
- If they have the same weight, then group C contains the heavier egg; otherwise, the heavier of A and B contains the heavier egg.
- Take the heavier group of 3 eggs, weigh any two eggs. If they have different weight, we can tell which egg is heavier. If they have the same weight, then the remaining egg is heavier.
- In total we have weight two times.

(b) In this case, when we have two groups of different weight, we do not know which groups is the abnormal one because we do not know whether the abnormal one is heavier or lighter. However, we will know that the third group is normal in such case.

We can revise our algorithm as follows:

- Take groups A and B to weigh.
- If they are the same, we know the abnormal egg is in group C. Then we weigh group C with group A and we will find out whether the abnormal egg is heavier or lighter (because group A is normal).
- Otherwise, we know the normal egg is in either group A or group B. Then we weigh group A with group C: if same, this means group B contains the abnormal egg and we can also tell whether it is heavier or lighter by the result of the first weigh.
- This means that using 2 weighs, we know the abnormal group and we know whether it is heavier or lighter than normal.
- Then we can use 1 weigh as in 4a to find out the abnormal egg in the abnormal group.
- In total we have weight three times.