## Baccalaureat exercise

We consider a large file which contains the marks of all the students at the Baccalaureat, in all the disciplines (one line gives for one student his mark in a discipline).

A line has the following format: 12345 french 14

The first field is the student ID, the second field is the discipline and the third field is the mark. We assume that each student has 10 marks and the the file is consistent (no missing marks).

We suppose we have in a Spark program a RDD variable initialized with the input file :

```
JavaRDD<String> data = sc.textFile(inputFile);
```

**Question 1**
Give the Spark program which computes the average mark  for each student.

**Question 2**
We now suppose that students have a variable number of marks (as they may have optional courses). This implies that the number of marks for each discipline is also variable. Give the program which computes the average in each discipline.

## Web log exercise

We consider a large file which contains the access logs of a web site. This file includes one line for each access with the following format:    pageId  clientIP

You can extract the nth element from each line l with: l.split(" ")[n]

We suppose we have in a Spark program a RDD variable initialized with the input file :

```
JavaRDD<String> data = sc.textFile(inputFile);
```

**Question 1**
Give the Spark program which computes the total number of access for each page

**Question 2**
Then, add the code which computes the percentage of access for each page (with respect to the total of accesses)

**Question 3**
Then, add the code to find the pageId with the highest percentage and print it

**Question 4**
Give the Spark program which computes for each client the most accessed page
Indication : you can create a JavaPairRDD ((pageId, clientIP), 1) then addition the 1 for each page-client. Then, you can transform it into a JavaPairRDD (clientIP, (pageid, total)) and then do a reduceByKey which keeps for each client only the page with the maximal total.

## Store management exercise

We consider a large file which contains the sales (here called transactions) from a set of stores. This file includes one line for each transaction with the following format:

**storeid productid number totalprice**

- storeid : the identifier of the store
- productid : the identifier of the product
- number : the number of products sold in the transaction
- price : the total price of the transaction

All these fields are Integers.
You can extract the nth element from each line l with: `l.split(" ")[n]`

We suppose we have in a Spark program a RDD variable initialized with the input file :

```
JavaRDD<String> data = sc.textFile(inputFile);
```

Below, we give (indication) the required number of lines needed to answer the question.

**Question 1**
Provide the Spark code which computes for each product the total number of sales (less than 10 lines)

**Question 2**
Extend the previous code to print the productid of the best-selling product (less than 5 lines)
Indication : you can use methods SortByKey() et take(n) described in the lecture

**Question 3**
Provide the Spark code which computes for each product the number of stores where the product is sold (less than 10 lines)
Indication : any RDD provides the distinct() method which removes duplicates