Big Data: Architectures and Data Analytics

February 2, 2022

Student ID _____

First Name _____

Last Name _____

The exam is **closed book** and lasts **2 hours**

Part I

Answer the following questions. There is only one right answer for each question.

Report your answers in the following boxes.

	Question 1	Question 2
Answer		

1. (2 points) Consider the input HDFS folder *myFolder* that contains the following two files:

   - NamesItaly.txt
     - the text file NamesItaly.txt contains the following four lines
       Paolo,Turin
       Luca,Rome
       Luca,Florence
       Paola,Turin
   - NamesFrance.txt
     - the text file NamesFrance.txt contains the following two lines
       Paolo,Nice
       Luis,Paris

   Suppose that you are using a Hadoop cluster that can potentially run up to 10 instances of the mapper class in parallel. Suppose the HDFS block size is 128MB. Suppose to execute a MapReduce application for Hadoop that analyzes the content of myFolder. Suppose the map phase emits overall the following 3 key-value pairs, associated with the input lines starting with "P" (output map phase/input reduce phase):

   ("Turin", 1)
   ("Turin", 1)
   ("Nice", 1)

Suppose the number of instances of the reducer class is set to 5 and suppose the reduce method of the reducer class sums the values associated with each key and emits one pair (city, NullWritable) for the keys associated with a sum less than 2. Specifically, suppose the following 1 pair is overall emitted by the reduce phase:

("Nice", NullWritable)

Considering all the instances of the reducer class, overall, how many times is the ***reduce method*** invoked?

a) 5

b) 3

c) 2

d) 1

2. (2 points) Consider the following Spark Streaming application.

```
import ..;
public class SparkDriver {
        public static void main(String[] args) throws InterruptedException {
                SparkConf conf = new SparkConf().setAppName("Spark Streaming - Question");

                // Create a Spark Streaming Context object
                JavaStreamingContext jssc = new JavaStreamingContext(conf,
                                                                Durations.seconds(10));

                // Define a DStream associated with the TPC socket localhost:9999
                JavaDStream<String> inputDStream = jssc.socketTextStream("localhost", 9999);

                /* Part A */
                // Map the input strings to integers
                JavaDStream<Integer> inputAIntDStream = inputDStream
                                .map(value -> Integer.valueOf(value));

                // Compute max
                JavaDStream<Integer> maxADStream = inputAIntDStream
                                .reduce((v1, v2) -> Math.max(v1, v2));

                // Define windows - windowDuration=30s - slideDuration=10s
                JavaDStream<Integer> maxAWinDStream = maxADStream
                                .window(Durations.seconds(30), Durations.seconds(10));

                // Compute max again
                JavaDStream<Integer> resADStream = maxAWinDStream
                                .reduce((v1, v2) -> Math.max(v1, v2));

                // Store the result of Part A
                resADStream.dstream().saveAsTextFiles("outputPartA", "");
```

```
                /* Part B */
                // Map the input strings to integers
                JavaDStream<Integer> inputBIntDStream = inputDStream
                                .map(value -> Integer.valueOf(value));

                // Compute max
                JavaDStream<Integer> maxBDStream = inputBIntDStream
                                .reduce((v1, v2) -> Math.max(v1, v2));

                // Define windows - windowDuration=30s - slideDuration=10s
                JavaDStream<Integer> resBDStream = maxBDStream
                                .window(Durations.seconds(30), Durations.seconds(10));

                // Store the result of Part B
                resBDStream.dstream().saveAsTextFiles("outputPartB", "");

                /* Part C */
                // Define windows - windowDuration=30s - slideDuration=10s
                JavaDStream<String> inputCWinIntDStream = inputDStream
                                .window(Durations.seconds(30), Durations.seconds(10));

                // Map the input strings to integers
                JavaDStream<Integer> inputCIntDStream = inputCWinIntDStream
                                .map(value -> Integer.valueOf(value));

                // Compute max
                JavaDStream<Integer> resCDStream = inputCIntDStream
                                .reduce((v1, v2) -> Math.max(v1, v2));

                // Store the result of Part C
                resCDStream.dstream().saveAsTextFiles("outputPartC", "");

                // Start the computation
                jssc.start();                   jssc.awaitTerminationOrTimeout(120000);
                jssc.close();
        }
}
```

Which one of the following statements is true?

a) Independently of the content of inputDStream, **resADStream**, **resBDStream**, and **resCDStream** contain the same integer values.

b) Independently of the content of inputDStream, **resADStream** and **resBDStream** contain the same integer values, while **resCDStream** contains different integer values with respect to **resADStream** and **resBDStream**.

c) Independently of the content of inputDStream, **resADStream** and **resCDStream** contain the same integer values, while **resBDStream** contains different integer values with respect to **resADStream** and **resCDStream**.

d) Independently of the content of inputDStream, **resBDStream** and **resCDStream** contain the same integer values, while **resADStream** contains different integer values with respect to **resBDStream** and **resCDStream**.

Part II

PoliApps is a marketplace that is used to sell mobile apps. PoliApps manages millions of apps and is used by millions of users. PoliApps computes statistics about the usage of its apps and the characteristics of its users. The analyses are based on the following input data sets/files.

- Apps.txt
  - Apps.txt is a text file containing the list of mobile apps available on PoliApps. Each line of Apps.txt is associated with one app. PoliApps manages millions of apps.
  - Each line of Apps.txt has the following format
    - AppId,AppName,Price,Category,Company

      where *AppId* is the unique identifier of the app while *AppName* is its name, *Price* is its price, *Category* is its category (game, office, finance, etc.), and *Company* is the company that developed the app.

    - For example, the following line

      App10,PolitoApp,0,Education,Polito

      means that the app identified by the AppId ***App10*** is called PolitoApp, it costs ***0*** euros, it belongs to the ***Education*** category, and it is developed by ***Polito***.

- Users.txt
  - Users.txt is a text file containing the profiles of the users of PoliApps. Each line is associated with one user. PoliApps has millions of users.
  - Each line of Users.txt has the following format
    - UserId,Name,Surname

      where *UserId* is the unique identifier of the user while *Name* and *Surname* are his/her name and surname, respectively.

    - For example, the following line

      User15,Paolo,Garza

      means that the name and surname of the user identified by the id ***User15*** are Paolo and Garza, respectively.

- Actions.txt
  - Actions.txt is a text file that is used to track downloads, installations, and removals of apps. A new line is appended at the end of Actions.txt every time a user downloads, installs, or removes an app. Actions.txt stores more than 20 years of data.
  - Each line of Actions.txt has the following format
    - UserId,AppId,Timestamp,Action

      where *UserID* is the identifier of the user who performed the action specified in the field *Action* on the app with id *AppId* at time *Timestamp*. *Action* can assume one of the following three values: "Download", "Install", or "Remove". Pay attention that the same user can perform the same action on the same app multiple times, in different timestamps. For instance, a user can install the same app multiple times. The field *Timestamp* is a string and its format is "YYYY/MM/DD-HH:MM:SS".

    - For example, the following line

      User15,App10,2019/01/01-23:01:15,Install

      means that ***User15*** installed (Action is equal to ***Install***) the app ***App10*** on ***January 1, 2019,*** at ***23:01:15***.

Exercise 1 – MapReduce and Hadoop (8 points)

Exercise 1.1

The managers of PoliApps are interested in performing some analyses about their apps.

Design a single application, based on MapReduce and Hadoop, and write the corresponding Java code, to address the following point:

1. *Companies with mainly free apps in the category Game.* The application considers only the apps of the category Game (Category='Game') and selects the companies that developed a number of free apps greater than the number of non-free apps, considering only the category Game. An app is free if its price is equal to 0. If the price of an app is greater than 0 then the app is non-free. Store the selected companies and the number of apps of the category Game developed by each of the selected companies in the output HDFS folder (one pair (company, number of developed apps of the category Game) per output line).

Suppose that the input is Apps.txt and has been already set. Suppose that also the name of the output folder has been already set.

- Write only the content of the Mapper and Reducer classes (map and reduce methods. setup and cleanup if needed). The content of the Driver must not be reported.

- Use the following two specific multiple-choice questions (**Exercises 1.2 and 1.3)** to specify the number of instances of the reducer class for each job.
- If your application is based on two jobs, specify which methods are associated with the first job and which are associated with the second job.
- If you need personalized classes, report for each of them:
    - the name of the class
    - attributes/fields of the class (data type and name)
    - personalized methods (if any), e.g., the content of the toString() method if you override it
    - do not report the get and set methods. Suppose they are "automatically defined"
- **Write your code on your papers.**

Answer the following two questions to specify the number of jobs (one or two) and the number of instances of the reducer classes.

Exercise 1.2 - Number of instances of the reducer - Job 1

Select the number of instances of the reducer class of the first Job

- (a) 0
- (b) exactly 1
- (c) any number >=1 (i.e., the reduce phase can be parallelized)

Exercise 1.3 - Number of instances of the reducer - Job 2

Select the number of instances of the reducer class of the second Job

- (a) One single job is needed
- (b) 0
- (c) exactly 1
- (d) any number >=1 (i.e., the reduce phase can be parallelized)

Exercise 2 – Spark and RDDs (19 points)
The managers of PoliApps are interested in performing some analyses about their apps.

The managers of PoliApps asked you to develop one single application to address all the analyses they are interested in. The application has five arguments: the input files Apps.txt, Users.txt, and Actions.txt, and two output folders "outPart1/" and "outPart2/", which are associated with the outputs of the following points 1 and 2, respectively.

Specifically, design a single application, based on Spark, and write the corresponding code, to address the following two points:

1. *Apps with more monthly installations than monthly removals in all the twelve months of the year 2021.* This first part of the application considers only the data related to the year 2021 and selects the apps with a number of monthly installations greater than the number of monthly removals in each of the twelve months of the year 2021. The number of monthly installations of an app in a specific month is the number of lines of Actions.txt associated with that app and that month for which Action is equal to 'Install'. Similarly, the number of monthly removals of an app in a specific month is the number of lines of Actions.txt associated with that app and that month for which Action is equal to 'Remove'. The pairs (AppId, AppName) associated with the selected apps are stored in the first HDFS output folder (one pair per line).

2. *Apps with the maximum number of new users after December 31, 2021.* This second part of the application selects the apps characterized by the maximum number of distinct new users after December 31, 2021 (a user is considered a new user of an app after December 31, 2021, if that user installed that app after December 31, 2021, and never installed the same app before January 1, 2022). The AppIds of the selected apps are stored in the second HDFS output folder (one AppId per output line).
   Pay attention. All the apps associated with the maximum number of new users after December 31, 2021, are stored in the output folder (one AppId per output line).

   Examples Point 2

   - *First example.* For the sake of clarity, suppose that there are only three apps: App1, App2, and App3. Suppose that (i) App1 was installed by 100 new users after December 31, 2021, (ii) App2 was installed by 130 new users after December 31, 2021, and (iii) App3 was installed by 68 new users after December 31, 2021. In this first example, the id **App2** will be stored in the second output folder.
   - *Second example.* For the sake of clarity, suppose that there are only three apps: App1, App2, and App3. Suppose that (i) App1 was installed by 130 new users after December 31, 2021, (ii) App2 was installed by 130 new users after December 31, 2021, and (iii) App3 was installed by 68 new users after December 31, 2021. In this second example, the ids **App1** and **App2** will be stored in the second output folder (one AppId per line).
   - Pay attention that the actual input file contains millions of apps.

   - **Write your code on your papers.**

   - You do not need to report imports. Focus on the content of the main method.

- Suppose both **JavaSparkContext sc** and **SparkSession ss** have been already set.

- If you need personalized classes, report for each of them:

    - the name of the class
    - attributes/fields of the class (data type and name)
    - personalized methods (if any), e.g., the content of the toString() method if you override it
    - do not report the get and set methods. Suppose they are "automatically defined"