**1. Overview**

The aim of this test is to get an accurate picture on coding skills and understand the thought process of the candidates to arrive at a solution.

Please do not hesitate to seek any clarification at any given point of time.

**2. Description / Scenario**

Educational games are played at school (in groups) and at home (individually) by pre-schoolers continuously. A Gaming Company (GC) provide playset–GC school edition and GC home edition. The Company is planning to offer a discount scheme based on the number of hours a student plays.

The scheme is as follows: If played at school for 5 hours, a student gets a discount time of 15 minutes (5% of total time), and if played at home for 5 hours, he/she gets a discount time of 30 minutes (10% of total time). As the subscription fee for the school edition playset is cheaper than the home edition one, the hours offered on discount for the home edition playset is more.

The company will calculate the discount based on the discount hours, every month. This calculation is done taking the **'Input\_Start\_Students\_Discounts.txt'** discount hours file for all students, and a **'Input\_Game\_Events.json'** file for all game play events emitted from games as input, applying events on the 'Input\_Start\_Students\_Discounts’ file based on the discount type (Discount for School: DS, Discount for Home: DH) and playset type (School: S, Home: H), and computing the 'Output\_End\_Students\_Discounts' discount hours. Depending on the playset used in the daily game events file, each event is to be recorded as 'debit' or 'credit' into school and home editions in the “Input\_Start\_Students\_Discounts” file. (Note that, ‘Input\_Game\_Events.json’ , a json file  is parsed and transformed to a csv file before applying it to ‘Input\_Start\_Students\_Discounts.txt’ csv file.)

The student play games on mobile tablets and playsets. The play activities are logged and sent as json format. Each activity may have multiple rounds. In a single round, a student is prompted for a letter on screen and must place same letter piece in playset. This activity of identifying the correct letters in game rounds is logged as an event in JSON format in a ‘daily game events’ file. Refer Data dictionary and input files to understand data.

**Input:**

Input\_Start\_Students\_Discounts.txt File: contains 'start' discount minutes of all students (this file is created at the end of the previous day). Below is an example on data format of csv file

|  |  |  |  |
| --- | --- | --- | --- |
| StudentId | PlaysetNumber | PlaysetType | DiscountMinutes |
| AA000100 | CHN000100 | S | 1.5 |
| AA000100 | CHN000200 | H | 0.3 |

Input\_Game\_Events.json File: contains game play events for all students daily (this file is generated by another batch process at end of day with all daily events in json format). Below is an example on data format of json file

{events: [ {

“eventId”: 2789

"studentId": “MF200”,

"gameCode": "e2t1",

"playsetType": "H",

"startTime": "2019-04-01T10:11:02.000Z",

"endTime": "2019-04-01T10:15:20.000Z",

"data": [

{

"round": 1,

"roundStartTime": "2019-04-01T10:11:55.000Z",

"roundEndTime": "2019-04-01T10:13:10.000Z",

"targetText": ['M'],

"responseText": ['M']

}

]

},

{……….},

{……….}

] }

**Process to build:**

**Step 1:** Read ‘Input\_Game\_Events.json’ files, transform each event as a new row with new columns and write in a new ‘Output\_Game\_Events \_Discounts.txt’ file in csv format. This new file columns are:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| EventId | StudentId | EventType | PlayMinutes | NoOfResponses | NoOfSuccessfulResponses |

The logic to fill value for these columns is:

EventId: eventId field from Events source file

StudentId: studentId field from Events source file

EventType: “DH” if playsetType = “H" from Events source file

“DS” if playsetType = “S" from Events source file

PlayMinutes: Calculate and project total time the student played in all rounds (round 1 played time + round 2 played time………. till n rounds)

NoOfResponses: responseText field in events file is an array with length equal to the number of responses against a targetText prompted to student and content of field are the actual letters responded while playing the game. For example, student is prompted to put letter “S” on screen and student puts letter “M” piece in playset, again gets a prompt to lift M and place S, this time places S correctly. So, the targetText field is [“S”] and responseText field is [“M”,” S”]. Based on this find the total number of responses in all rounds of an event record and project it.

NoOfSuccessfulResponses: Find the number of successful responses student makes to find and place correct letter in playset across all rounds. In case there is no successful response, project 0 in the field.

**Step 2:** Read Start of Day ‘Input\_Start\_Students\_Discounts.txt’ and ‘Output\_Game\_Events \_Discounts.txt’ files generated as part of Step 1 problem, compute new discount minutes and write in new ‘Output\_ End\_Students\_Discounts.txt’ discount minutes file.

For each game event in the ‘Event’ file,

Apply event into matching student records in the discount file

If Event Type =DS,

                               For PlaysetType =S, DiscountMinutes = DiscountMinutes + PlayMinutes

                               For PlaysetType =H, DiscountMinutes = DiscountMinutes - PlayMinutes

If Event Type =DH,

                                For PlaysetType =S, DiscountMinutes = DiscountMinutes - PlayMinutes

                                For PlaysetType =H, DiscountMinutes = DiscountMinutes + PlayMinutes

**In case the calculated discount minutes are negative, it is fine.**

(The logic behind this calculation is that the company **does not want** to provide a discount on both home and school playset hours, but it is a differential discount. If discount is offered for school hours, the same proportion of discount is removed from any hours on the home playset, and vice-versa.)

The output file columns are:

|  |  |  |  |
| --- | --- | --- | --- |
| StudentId | PlaysetNumber | PlaysetType | DiscountMinutes |

Along with the generation of output file, also **find the student ids with minimum and maximum discount minutes in the program.**

**Attached files:**

Input#

1. Input\_Start\_Students\_Discounts.txt (csv format, comma delimited, 1st line is header line)

2. Input\_Game\_Events.json (json format)

**Assumptions:**

1. Assume file sizes as small enough in Megabytes to fit in memory.
2. Assume ‘Daily Event’ file as being much bigger as compared to discount files.
3. Work on a single thread model

**Expected Action:**

1. Write a program code in Java /Python/Scala or any other language for single thread model to build the Step 1 process. Use the JSON parsing library of your choice.

2. Write a program code in Java /Python/Scala or any other language for single thread model to build the Step 2 process. Step 2 has its input file from Step 1. Along with the output file, **also print the student ids with minimum and maximum discount minutes in the program.**

3. State any other specific assumptions, **if made.**

4. Incorporate error handling in code.

5. Explain in short, if any optimizations have been applied using code comments.

6. Assume the file sizes are not in MBs but in GBs. The game events file size = 60GB, start\_discount file size = 2 GB. There is a cluster of 3 nodes with RAM size = 8GB, disk size = 100 GB. If the above problem of Step 2 to find student ids with minimum and maximum discount minutes is to be worked with a parallel distributed computing framework, then explain the below **in short**:

a. Approach and steps on building Step 2 problem using distributed computing programming model like MapReduce. (Kind of short pseudo code is also ok) What are the advantages/disadvantages, if any, of your approach.

b. What are the distributed computing frameworks which could be used to build overall data pipeline for both Step 1 and Step 2 as a batch process? Explain why to choose or why not on the frameworks. Does the available cluster memory (RAM/disk size) fulfil memory requirements or requires a tweaking ? Explain memory usage with the framework you choose.

c. If the Input\_Game\_Events.json file is not a file but real time stream of data produced and Input\_Start\_Students\_Discounts.txt is a file refreshed daily, what will be the approach to build the same process of both Step 1 and Step 2 as a real time data pipeline to show students with minimum/maximum discount minutes every 15 minutes on a portal/dashboard? Explain on the stages in the data pipeline. Use appropriate stream producer/consumer framework.

**Data Dictionary:**

|  |  |
| --- | --- |
| **Input\_Game\_Events Json File** | |
| EventId | Id of event record |
| StudentId | Unique id of the student |
| GameCode | DS (Discount at School) / DH (Discount at Home) |
| PlaysetType | Playset type S (School) / H (Home) |
| StartTime | Start Time of event |
| EndTime | End Time of event |
| Round | Round Number of a game play event |
| RoundStartTime | Starting time for the round in an event |
| RoundEndTime | End time for the round in an event |
| TargetText | Letter prompted to student in a round of game event |
| ResponseText | Letters answered by student in a round of game event |
|  |  |
| **Input\_Start\_Students\_Discounts csv File** | |
| StudentId | Unique id of the student |
| PlaysetNumber | Playset Edition Number |
| PlaysetType | Playset type S (School) / H (Home) |
| DiscountMinutes | discount minutes (used in discount calculation) |
|  |  |
| **Output\_Game\_Events\_Discounts (Step 1 Output csv file)** | |
| EventId | Unique Id of the event record |
| StudentId | Unique id of the student |
| EventType | DS (Discount at School) / DH (Discount at Home) |
| PlayMinutes | Minutes spent in all rounds in an event. |
| NoOfResponses | Total number of responses in all rounds of an event |
| NoOfSuccessfulResponses | number of correct responses where student places correct letter across all rounds in an event |
|  |  |
| **Output\_ End\_Students\_Discounts (Step 2 Output csv file)** | |
| StudentId | Unique id of the student |
| PlaysetNumber | Playset Edition Number (Home/School) |
| PlaysetType | Playset type S (School) / H (Home) |
| DiscountMinutes | discount minutes updated from start of day file and  the transformed daily game events file |