## TaskA

The algorithm filters input data to extract names and hobbies of people with "Armenia" as their nationality using a mapper-only Hadoop job. It processes each line, checks the nationality field, and outputs matching records. This is a map-only job, which means that there is no reducer phase. The reason is that there are no groupings or aggregations, meaning that each record can be processed separately, without interacting with other records.

## TaskB

This task requires two map-reduce jobs:
The TaskB job focuses on counting the page accesses and identifying the top 10 most accessed pages. In the PageAccessMapper, the program reads data from a file named access_logs.csv, extracting the pageId from each line. It then emits a key-value pair, where the pageId is the key and the value is 1. The reducer, PageAccessReducer, aggregates these counts, storing the results in a priority queue to track the top 10 pages. The queue is configured to discard pages with the least accesses once the queue exceeds 10 entries, ensuring that only the top pages are retained. Finally, the top 10 pages and their counts are written to the output.
In the PageDetailsJoin job, the aim is to join the details of the top 10 pages with additional information stored in a separate file, pages.csv. This job uses two mappers: the first (TopPagesMapper) processes the output from TaskB by emitting the page pageId and its access count, while the second (PagesMapper) reads the pages.csv file and extracts the page details, including the name and nationality for each page. The reducer, JoinReducer, combines these two datasets based on the common pageId key. It outputs the top pages with their name and nationality, ensuring that only pages that were identified as top 10 in TaskB are included in the final output. The use of MultipleInputs allows the job to handle both input files in a single job, facilitating the join process across different data sources.

## TaskC

The TaskC job is a MapReduce job with both map and reduce phases.
In the mapper phase (FacebookPageCountMapper), the job processes input and filters rows based on a specific country code (from the 4th column). If the country code is a valid number between 1 and 50, it emits the country code as the key and a constant value of 1 as the value, indicating that this entry is related to a Facebook page from that country. This is done for each valid record in the input.
In the reducer phase (FacebookPageCountReducer), the job aggregates all the counts for each unique country code. The reducer iterates over the values (which are all 1s) associated with each country code and sums them up, effectively counting how many Facebook pages are related to each country. The final output is a list of country codes and their respective counts.

**TaskD**

The TaskD job is a complete MapReduce job that involves both map and reduce phases. It processes data from two different input sources, using two distinct mappers. The first mapper, MyPageMapper, handles the first input, extracting page IDs and emitting them with a zero count, marking their presence for later processing. The second mapper, ConnectednessFactorMapper, processes the second input, which contains relationships between pages and their friend IDs, and emits each friend ID with a value of one. The reducer, ConnectednessFactorReducer, then aggregates the counts for each friend ID, summing the 1s emitted by the second mapper to calculate the total number of connections or interactions for each friend ID. This design makes the job not a map-only job, as it combines the results from both mappers in the reduce phase to provide the final output. The use of MultipleInputs allows the job to handle data from two separate sources, merging and aggregating the information during the job execution.

**TaskE**

In TaskE, the aim is to analyze user access logs and detect their favorite pages based on the frequency of their visits. This is a map-reduce job, because counting is needed. In the mapper, each log entry is processed by extracting the user ID and the page ID from the log. This creates key-value pairs where the key is the user ID, and the value is the page ID that the user visited. The mapper essentially organizes the data by user, so we can later analyze their activity.

The reducer takes this organized data and performs the aggregation. For each user, it calculates two things: the total number of page visits (how many times the user has accessed a page) and the number of unique pages they've visited (how diverse their interests are). It then outputs the user ID along with these two metrics: the total visit count and the count of distinct pages.

**TaskF**

In TaskF, the goal is to identify friends who have never accessed a page in the system. The process involves two mappers and a reducer. The first mapper, FriendshipMapper, processes friendship data by mapping a person's ID to their friends. It reads the data line by line, extracting two person IDs for each friendship, and generates key-value pairs with the person’s ID as the key and their friend’s ID as the value. The second mapper, AccessLogMapper, processes the access log data. For each user access record, it maps the user’s ID to the page they’ve accessed, generating key-value pairs with the user ID as the key and the page ID as the value.

The FriendNotAccessedReducer then collects the data from both mappers for each person, creating lists of their friends and the pages they’ve accessed. It checks each friend and identifies those who haven’t accessed any pages, outputting a list of these friends. The task essentially combines two datasets — one for friendships and one for access logs — to help pinpoint friends who haven’t engaged with the content, highlighting users who may be more passive in the system.

**TaskG**

This task is completed using two map-reduce jobs. The AccessLogMapper processes each access log entry by extracting the user ID and access time, outputting them as key-value pairs where the user ID is the key and the access time is the value. The AccessLogReducer then checks the latest access time for each user and compares it with the timestamp for 14 days ago. If a user hasn't accessed anything in that time frame, they are flagged as "disconnected." This helps identify inactive users in the system. On the other hand, ExtractMatchingIDs is focused on filtering user data based on a list of valid IDs. The IDFilterMapper loads these IDs from a file and checks whether each user in the input data matches any of the valid IDs. If they do, it outputs their user ID and name. This job doesn't need a reducer because it’s simply extracting matching records. Both jobs efficiently process data, with TaskG identifying inactive users based on access logs and ExtractMatchingIDs filtering out users based on predefined valid IDs.

**Task H**

This map-reduce job calculates the number of friends for each user and the average number of friendships across all users. The mapper counts friends per user and tracks total friendships and user counts, while the reducer aggregates these counts and computes the average friendships per user.