Suppose there is a website tracking user activities to prevent robotic attack on the Internet. Examples are in the “data.txt”. Design an algorithm to identify user IDs that have more than 500 activities within any given 10 minutes.

In order to design the algorithm, there are a few factors I need to consider. 1) *What the data looks like?* The example shows that users come in a sequential order, i.e., the time stamp is in an ascending order. It resembles a list where the length of the list is not specified but keeps growing. 2) *Large data set.* Presumably there will be thousands or millions of users (rows) in this table, and their IDs are non-unique and unordered. In the big data context, it will be inefficient to sort the table by user IDs. Conversely, it is better to implement the above list in a data structure that can map a unique user ID (key) to its associated activity count (value), which will speed up the searching process. Hence, a hash table can be used as such a data structure.

Based on the above scenarios, I designed the following algorithm:

1. Read the file line by line to *avoid* reading the entire file (because it is large and the task can be accomplished incrementally);
2. Start from the second line (because the first line is the header, and the data starts from the second line);
3. After reading in the new line, convert the three variables (user id, time stamp, activity count) to integers (it is easy to convert the 1st and 3rd but tricky for the 2nd variable, as it comes in a *HH:MMam* format. Therefore, a separate function is needed to convert the time to minutes);
4. Store the three variables in a single node, and add the node to the end of a linked list. Update the hash table based on the linked list, where the table key is the user id, and value corresponds to its activity count.
5. Keep summing up count values for the same user in the hash table (because if the same user comes to the website again in a later time and has additional activities, then I need to add the current count value to his original value);
6. If the time window exceeds 10min, remove the counts of first few nodes from the hash table until the window is within 10min. Therefore, the value of each key is updated by subtracting counts. Next remove those nodes from the linked list (because I need to reset the timer, i.e., the beginning of the window is supposed to move down from the linked list. It is possibly needed to delete first few nodes until the window is within 10min);
7. Print out the user id (key) corresponding to the last node of the linked list, if its count (value) is more than 500.
8. Repeat c) – g) in a loop until the last line of the file is read.

I implemented the above algorithm in Java, and saved the programs and output in the attached file. Please note that “WebTracker.java” is the main program, and output is saved in “output.txt”. In the current program, I used the file name “data.txt” by copying the file to Eclipse, and you can change it to “/your directory/data.txt” or “/your directory/your data”.