import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Date;

public class EmployeeAnalyzer {

public static void main(String[] args) {

String filename = "your\_input\_file.txt"; // Replace with the actual file name

analyzeEmployeeData(filename);

}

private static void analyzeEmployeeData(String filename) {

try (BufferedReader reader = new BufferedReader(new FileReader(filename))) {

String line;

String[] headers = null;

int consecutiveDaysCount = 0;

Date previousTimeOut = null;

double currentShiftHours = 0;

while ((line = reader.readLine()) != null) {

String[] data = line.split("\t");

if (headers == null) {

headers = data;

continue; // Skip header row

}

String name = data[headersListIndexOf(headers, "Employee Name")];

String positionId = data[headersListIndexOf(headers, "Position ID")];

String timeOutStr = data[headersListIndexOf(headers, "Time Out")];

SimpleDateFormat dateFormat = new SimpleDateFormat("MM-dd-yyyy hh:mm a");

Date timeOut = dateFormat.parse(timeOutStr);

// Check for consecutive days

if (previousTimeOut != null && daysBetween(previousTimeOut, timeOut) == 1) {

consecutiveDaysCount++;

} else {

consecutiveDaysCount = 1;

}

// Check for less than 10 hours between shifts but greater than 1 hour

if (previousTimeOut != null && hoursBetween(previousTimeOut, timeOut) < 10 && hoursBetween(previousTimeOut, timeOut) > 1) {

System.out.println(name + " (" + positionId + ") has less than 10 hours between shifts on " + timeOutStr);

}

// Check for more than 14 hours in a single shift

currentShiftHours += hoursBetween(previousTimeOut, timeOut);

if (currentShiftHours > 14) {

System.out.println(name + " (" + positionId + ") has worked for more than 14 hours on " + timeOutStr);

}

// Update previousTimeOut for the next iteration

previousTimeOut = timeOut;

// Reset consecutiveDaysCount and currentShiftHours if the streak is broken

if (consecutiveDaysCount > 1) {

consecutiveDaysCount = 0;

currentShiftHours = 0;

}

}

} catch (IOException | ParseException e) {

e.printStackTrace();

}

}

private static int headersListIndexOf(String[] headers, String columnName) {

for (int i = 0; i < headers.length; i++) {

if (headers[i].equals(columnName)) {

return i;

}

}

return -1;

}

private static long daysBetween(Date startDate, Date endDate) {

long difference = endDate.getTime() - startDate.getTime();

return difference / (24 \* 60 \* 60 \* 1000);

}

private static double hoursBetween(Date startDate, Date endDate) {

long difference = endDate.getTime() - startDate.getTime();

return difference / (60 \* 60 \* 1000.0);

}

}