'use strict';

import { WriteStream, createWriteStream } from "fs";

process.stdin.resume();

process.stdin.setEncoding('utf-8');

let inputString: string = '';

let inputLines: string[] = [];

let currentLine: number = 0;

process.stdin.on('data', function(inputStdin: string): void {

inputString += inputStdin;

});

process.stdin.on('end', function(): void {

inputLines = inputString.split('\n');

inputString = '';

main();

});

function readLine(): string {

return inputLines[currentLine++];

}

/\*

\* Complete the 'GetRejectedRequests' function below.

\*

\* The function is expected to return an INTEGER\_ARRAY.

\* The function accepts following parameters:

\* 1. STRING\_ARRAY requests

\* 2. INTEGER limit\_per\_second

\*/

function GetRejectedRequests(requests: string[], limit\_per\_second: number): number[] {

// Write your code here

const ipRequests: { [key: string]: { [key: number]: number[] } } = {}; // { IP: { second: [request\_ids] } }

const rejectedRequestIds: number[] = [];

// Iterate through each request

for (let i = 0; i < requests.length; i++) {

const [requestId, ipAddress, timestampStr] = requests[i].split(' ');

const requestIdNum = parseInt(requestId, 10);

const timestamp = Math.round(parseInt(timestampStr, 10) / 1000); // Convert to seconds

// Initialize the dictionary for this IP if it doesn't exist

if (!ipRequests[ipAddress]) {

ipRequests[ipAddress] = {};

}

// Initialize the array for this second if it doesn't exist

if (!ipRequests[ipAddress][timestamp]) {

ipRequests[ipAddress][timestamp] = [];

}

// If the number of requests from this IP in the current second exceeds the limit, reject the request

if (ipRequests[ipAddress][timestamp].length >= limit\_per\_second) {

rejectedRequestIds.push(requestIdNum);

} else {

// Otherwise, accept the request by adding the requestId to the list for this second

ipRequests[ipAddress][timestamp].push(requestIdNum);

}

}

return rejectedRequestIds;

}

function main() {

const ws: WriteStream = createWriteStream(process.env['OUTPUT\_PATH']);

const requestsCount: number = parseInt(readLine().trim(), 10);

let requests: string[] = [];

for (let i: number = 0; i < requestsCount; i++) {

const requestsItem: string = readLine();

requests.push(requestsItem);

}

const limit\_per\_second: number = parseInt(readLine().trim(), 10);

const result: number[] = GetRejectedRequests(requests, limit\_per\_second);

ws.write(result.join('\n') + '\n');

ws.end();

}