Coin Organization

Objective
Give practice with sorting in C.

Story
The advertising scheme worked. Unfortunately, you have many more orders coming in than can be completed in a timely fashion. You will organize individuals by the amount of money they pay you. There are two denominations used in Gameland: tokens, and bills. All bills are equivalent in value and all tokens are equivalent in value. Due to some individuals on popular online forms the value of tokens with respect to the value of bills fluctuates quite often. We compare the two currencies by stating "***a*** tokens is worth ***b*** bills".

We know much money each person gave us in terms of number of tokens and number of bills. We just need to sort them based on the newest value ratio given.

Problem
Given a list of individuals and how much they paid followed by a value ratio for the currency, sort the individuals from GREATEST to LEAST in terms of paid value.

Input
Input will begin with a line containing 1 integers, n ($1 \leq n \leq 100{,}000$), representing the number of paying customers. The following n lines will each contain a description of a customer. The i-th of which contains a single name followed by two space separated non-negative integers, t_i and b_i, ($0 \leq t_i, b_i \leq 100{,}000$), representing the number of tokens and the number bills the i-th person paid respectively.

The last line of input will contain 2 integers a and b, ($1 \leq a, b \leq 100{,}000$), those value represent the exchange rate between the two currencies "***a*** tokens is worth ***b*** bills".

Each name will have at least one and at most 20 characters. Names will not contain whitespace characters.

Output
Output should contain n lines. Each line will contain a corresponding resulting handle based on the input.

Sample Input	Sample Output
5 John 3 10 Jacob 7 7 Rob 5 8 Nancy 4 5 Phil 11 4 10 13	Phil Jacob Rob John Nancy
3 Alice 4 8 Bob 9 2 Carol 11 0 6 7	Carol Alice Bob

Explanation

Case 1

10 tokens are worth 13 bills.
We can say that each token is worth 13 “units” and each bill is worth 10 “units”.

We will first find out how many “units” each person has

John has 3 tokens and 10 bills which equates to 3 * 13 + 10 * 10 units = 139 units
Jacob has 7 tokens and 7 bills which equates to 7 * 13 + 7 * 10 units = 161 units
Rob has 5 tokens and 8 bills which equates to 5 * 13 + 8 * 10 units = 145 units
Nancy has 3 tokens and 10 bills which equates to 4 * 13 + 5 * 10 units = 102 units
Phil has 3 tokens and 10 bills which equates to 11 * 13 + 4 * 10 units = 183 units

Phil has paid the most.
Jacob is second.
Rob is third.
John is fourth.
Nancy is fifth.

Case 2
6 tokens are worth 7 bills.
We can say that each token is worth 7 “units” and each bill is worth 6 “units”.

We will first find out how many “units” each person has

Alice has 4 tokens and 8 bills which equates to 4 * 7 + 8 * 6 units = 76 units
Bob has 9 tokens and 2 bills which equates to 9 * 7 + 2 * 6 units = 75 units
Carol has 11 tokens and 0 bills which equates to 11 * 7 + 0 * 6 units = 77 units

Carol has paid the most.
Alice is second.
Bob is third.

Hints

Convert to Units: It would be ideal to convert all the currency into some common currency. I don’t recommend using division due to floating point errors, but it can be done if you wish to convert all currency into bills or all currency into tokens.

Alternatively you can create some alternate currency that I refer to as the “unit”.

Suppose ***a*** tokens = ***b*** bills.
Let 1 token = ***b*** units. We then multiply both sides by ***a***.
a tokens = ***ab*** units. We then substitute.
b bills = ***ab*** units. We then divide both sides by ***b***.
1 bill = ***a*** units.

Thus we can convert to units by multiplication.

Precision: Avoid using ints to prevent overflow. An int can only store up to around 2 billion, but the units can be up to 10 billion. Use long long ints to allow for 64 bits of precision.

Use Structs: You should try to store the people into a struct that stores their units, name, etc.

Grading Criteria

- Good comments, whitespace, and variable names
    - 15 points
- Standard input output
    - 10 points
- Create a way to compare two individuals
    - 10 points
- Using a high precision data type (e.g. long long int) to store the information for a user.
    - 5 points
- Store the people in an array that gets sorted.
    - 10 points
- Programs will be tested on 10 cases
    - 5 points each

No points will be awarded to programs that do not compile using "gcc -std=gnu11 -lm".

Sometime a requested technique will be given, and solutions without the requested technique will have their maximum points total reduced. For this problem use the merge/quick sort. ***Without this programs will earn at most 50 points! Using a sort built into the c library will also receive at most 50 points.*** *You can use one of the sorts on webcourses to base your program.*

Any case that causes a program to return a non-zero return code will be treated as wrong. Additionally, any case that takes longer than the maximum allowed time (the max of {5 times my solutions time, 10 seconds}) will also be treated as wrong.

No partial credit will be awarded for an incorrect case.