1. Write a program that reads in three points describing the vertices of a triangle and computes the midpoint triangle they define, i.e. the triangle whose vertices are the three midpoints of the previous triangle. A typical run of your program should look like this:

Enter triangle vertices: (0,0) (0,1) (1,0)

Midpoint triangle verts: (0,0.5)(0.5,0.5)(0.5,0)

2. Create a struct to hold a fraction. The struct should have an integer numerator and an integer denominator member. Declare 2 fraction variables and read them in from the user. Write a function called multiply that takes both fractions, multiplies them together, and returns the result as a decimal number. You do not need to reduce the fraction to its lowest terms. Print the result of the multiplication of the 2 fraction variables.

3. Students who take the course must achieve an overall grade of 60 or higher to receive a pass certificate, with the added benefit of receiving $50 for an overall grade in the [G, 100] range and $20 for a grade in the [60, G) range. The instructor will also place the top K students in the course "Hall of Fame". In this problem, you are asked to write a program to help the teacher list the students in the Hall of Fame and count the total number of dollars given out.

**Input format.**

The input gives 3 integers on the first line, N (a positive integer up to 10 000 for the total number of students), G (an integer in the range [60,100] for the voucher grade cutoff described in the question), and K (a positive integer up to 100 and not exceeding N for the lowest ranking in the Hall of Fame). Each of the next N lines gives a student's account number (a string no more than 15 digits long without spaces) and overall grade (an integer in the range [0, 100]), separated by spaces. The question ensures that there are no duplicate account numbers.

**Output format.**

First output the total values issued in one line. Then output the rank, account number, and grade of the students who entered the Hall of Fame in non-ascending order by total grade, separated by 1 space. Note that students with the same score enjoy a tie for ranking, and when ranking is tied, the output is in alphabetically ascending order by account number.

**input:**

10 80 5

cy@zju.edu.cn 78

cy@pat-edu.com 87

1001@qq.com 65

uh-oh@163.com 96

test@126.com 39

anyone@qq.com 87

zoe@mit.edu 80

jack@ucla.edu 88

bob@cmu.edu 80

ken@163.com 70

**output:**

360

1 uh-oh@163.com 96

2 jack@ucla.edu 88

3 anyone@qq.com 87

3 cy@pat-edu.com 87

5 bob@cmu.edu 80

5 zoe@mit.edu 80

**（Optional）**4. On Sina Weibo, there is a "quiet follow", where a user quietly follows someone who does not appear on the user's follow list, but the system will push the tweets posted by the person they quietly follow to the user. Now let's be a network detective and find out who is likely to be followed by someone based on their follow list and their likes on other users.

**Input format.**

The input starts with a list of followers for a user given in the first line in the following format.

N user1 user2 ...... user\_N, where N is a positive integer not exceeding 5000 and each user i (i=1, ... , N) is the ID of the user being followed, which is a string of 4 digits in length consisting of numbers and letters, separated by spaces. After that, the information about the user's likes is given: first, a positive integer M not exceeding 10000, followed by M lines, each line giving the ID of the user whose likes are given and the number of likes on the user (not exceeding 1000), separated by spaces. Note: A user ID is a unique identifier for a user. The question ensures that there are no duplicate users in the followers list and no duplicate users in the likes information.

**Output format.**

We assume that people who have more likes than the average number of likes and are not on the user's follow list are likely to be quietly followed by the user. If there is no such person, then output "Bing Mei You".

**Input1:**

10 GAO3 Magi Zha1 Sen1 Quan FaMK LSum Eins FatM LLao

8

Magi 50

Pota 30

LLao 3

Ammy 48

Dave 15

GAO3 31

Zoro 1

Cath 60

**Output1:**

Ammy

Cath

Pota

**Intput2:**

11 GAO3 Magi Zha1 Sen1 Quan FaMK LSum Eins FatM LLao Pota

7

Magi 50

Pota 30

LLao 48

Ammy 3

Dave 15

GAO3 31

Zoro 29

**Output2:**

Bing Mei You

**（Optional）**5. There is a "like" function on Weibo, where you can like a blog post you like to show your support. Every blog post has some tags that portray its characteristics, and the type of blog post you like also indirectly portrays your characteristics. However, there is a type of person who likes everything they see to make their presence felt, and this type of person is called a "like fiend". The tags they like are so scattered that they don't have a distinct identity. This question asks you to write a program to find the top 3 like-lovers by counting the number of different tags each person likes.

**Input format.**

The input gives a positive integer N (≤ 100) in the first line, which is the number of users to be counted. In the next N rows, each row lists the hashtags of one user. The format is "Name K F1... FK", where Name is a non-empty user name with no more than 8 lowercase letters, 1 ≤ K ≤ 1000, Fi(i=1,⋯, K) is the number of the characteristic tag, and we will all characteristic tags from 1 to 107. We number all feature tags from 1 to 107. The numbers are separated by spaces.

**Output format.**

Count the number of different tags liked by each person, find the top 3 with the largest number, and output their usernames in a row in order, separated by 1 space, with no extra spaces at the end of the row. If there is a tie, the one with the smallest average number of tag appearances is output, and the title ensures that such users are not tied. If there are less than 3 users, the missing ones are filled in with -, for example, mike jenny - means there are only 2 users.

**input**.

5

bob 11 101 102 103 104 105 106 107 108 108 107 107

peter 8 1 2 3 4 3 2 5 1

chris 12 1 2 3 4 5 6 7 8 9 1 2 3

john 10 8 7 6 5 4 3 2 1 7 5

jack 9 6 7 8 9 10 11 12 13 14

**output.**

jack chris john