Mu-Sigma squared labs Entrance Examination

Mu-Sigma squared labs Entrance Examination

Assume that you are leading an exam conducting agency called *e-litmus*. A reputed research institution $\mu\sigma^2$ *labs* has granted you the task of conducting entrance examinations for admission in its labs for the position of *junior scientists*. Interested aspirants may write entrance examinations which are conducted on 15^{th} of every month (100 marks for each monthly exam, and results along with marks are declared on the 23^{rd} of the same month). An aspirant may write the exams multiple number of times, however **only a special averaged score** for her/him **will be kept** in the database (see the **explanation** below), along with an unique roll number (same as your IITM roll number format: Eg. CS14B003, AE15S401, CE13D111 etc.) assigned to her/him. Your task is to maintain a database of entrant information in the following format for an entry: <Roll-number, Current-updated-marks> in a list sorted on the basis of marks. The list needs to be kept sorted in descending order of marks (out of 100, and approximated up to 3 decimal places, see **note on fractional precision**) by 25^{th} of that month, immediately after scores are obtained. **In case of a tie** in the marks, sort the records based on **increasing order of sum of ASCII values of the roll number**. $\mu\sigma^2$ *labs* has stringent admission criteria: It seeks the sorted list of marks on 28^{th} of the month, and checks if $\geq \lfloor M\% \text{ of } S \rfloor$ top students have scored $\geq N\%$ of the top marks (See the **explanation** below). If so, it **provides admission to the top** $\lfloor 10\% \text{ of } S \rfloor$ **students**, where S is the current number of entries in the list. In that case, you have to **delete the entries for the admitted students** from the sorted list that *e-litmus* maintains. For subsequent months, updates (Eg. inserts) will be done on this list. This process can go on for K months. The idea of this task is to make use of **incremental sort** while facilitating **updates** on the current sorted list.

Explanation of selection criterion: Assume $M = 40, N = 80$. Let's say we have 20 students currently in the sorted (descending order of marks) list (say, for the month of July) and the top marks obtained is 78. If there are 8 (40% of 20) or more top students who have obtained 62.4 (80% of 78) or more marks, $\mu\sigma^2$ *labs* decides to give admission to top 10% of the students (in this case the top 2 students). Furthermore, record for those top 2 students have to be deleted from the list as well. If the criterion is not satisfied, no admission is given for that month.

Explanation on insertion of an entry: Assume you have the sorted list for the month of July and a student with roll number AE15B009 has written the exam for the month of August and obtained 62.5 marks. First check if the student has written the exam before, if not directly enter the record for AE15B009 in an appropriate location of the sorted list. If there is already a record <AE15B009, 73.5>, the record should be updated as <AE15B009, 68> (62.5+73.5/2) and re-positioned in the list. If the same student appears for the exam in the month of September and obtains 80.39 marks, the updated record should be <AE15B009, 74.195> (68+80.39/2). Furthermore, if AE15B009 rewrites the exam on October again (might not have been admitted despite an improvement in performance due to the admission criterion) and obtains only 50.493, the updated record will be <AE15B009, 62.344> (74.195+50.493/2). Note that after an update, the entry has to be re-positioned appropriately.

Note on fractional precision: std::setprecision(N) should do the task of expressing fractional precision. For example, refer:

#include<iostream>

#include<iomanip>

using namespace std;

int main() {

```
float x = 227.323543534543, y = 22;
cout << setiosflags(ios::fixed) << setprecision(3) << x << " " << y << endl;
cout << setprecision(1) << x << " " << y << endl;
cout << setprecision(30) << x << " " << y << endl;
return 0;
```

}

NOTE:

- You may change the specifications of incremental sort based on the previous assignment-4 / or you may come up with a new one, provided it follows proper Object-Oriented paradigms.
- While the sorting and selection tasks will be performed by *e-litmus* during the aforementioned timeline, your task is to design basic operational functionalities in the form of **queries** (as described later), which should be working at any point of the year irrespective of the timeline mentioned, i.e., the queries should work properly even if run in any week (say, first) of a month.

Input Format

P // P is the number of entries in the currently sorted list

<Roll_No_1 Marks_1>

⋮

<Roll_No_P Marks_P>

Q // Q is the number of queries to be made

<Query_1>

⋮

<Query_i>

⋮

<Query_Q>

where, <Query_i> can be any of the following:

INSERT <Roll_No Marks> // If an entry has not been **admitted** already (perform a check first!), this query will insert a **NEW** entry into the list and place it in an appropriate location. In case of an **ALREADY EXISTING ENTRY**, it will do the updating as discussed.

IS_ADMIT M N //If number of entries in the current list is not less than 10, this query will check the admission criteria as discussed and will pop out the students who got admission from the list in case of a success, else does nothing.

PRINT S E //This query will print all the students in the current list along with their marks, starting from index S till index E. If no indices S and E are included as arguments for the PRINT query, i.e. the query is simply given as PRINT, then the entire table must be printed. Assume, print index of list would start from 1 (not 0).

Constraints

$10 \leq P \leq 10^3$ // $P \geq 10$ is required only for **IS_ADMIT**, else it can be ≥ 1

$1 \leq Q \leq 10^3$

$M \leq 50$

$50 \leq N$

Output Format

For **INSERT**

If the entry has already been admitted, then the output should be "ALREADY ADMITTED"(in caps, with a space). Else, no output, only the appropriate action should be done.

For **IS_ADMIT**

i) If the admission criterion is satisfied, the output will be

YES

<Roll_No_1 Marks_1>

⋮

<Roll_No_L Marks_L> // where L is the number of top 10% students admitted.

ii) (If criterion is not satisfied) OR (If number of entries in the current list is less than 10) the output will be

NO

For **PRINT**

<Roll_No_S Marks_S>

<Roll_No_S+1 Marks_S+1>

⋮

<Roll_No_E-1 Marks_E-1>

<Roll_No_E Marks_E>

Sample Input 0

```
10
CS14S009 87.500
ME15B104 86.300
MM12B111 80.999
EE11D001 78.321
CS14S119 75.213
AE13D111 51.991
PH12B001 43.145
AM10M009 39.789
EP09B023 23.546
ME16D407 5.009
11
IS_ADMIT 40 90
PRINT 1 2
IS_ADMIT 50 80
PRINT 1 2
INSERT CE15B103 86.300
PRINT 1 4
INSERT AE15B009 73.500
INSERT AE15B009 62.500
INSERT AE15B009 80.390
INSERT AE15B009 50.493
PRINT
```

Sample Output 0

```
NO
CS14S009 87.500
ME15B104 86.300
```

```
YES
CS14S009 87.500
ME15B104 86.300
MM12B111 80.999
CE15B103 86.300
ME15B104 86.300
MM12B111 80.999
EE11D001 78.321
CE15B103 86.300
ME15B104 86.300
MM12B111 80.999
EE11D001 78.321
CS14S119 75.213
AE15B009 62.344
AE13D111 51.991
PH12B001 43.145
AM10M009 39.789
EP09B023 23.546
ME16D407 5.009
```