International Collegiate Programming Contest

The International Collegiate Programming Contest[1] (ICPC) is an annual programming competition held by the Association of Computing Machinery (ACM). Teams from universities all around the world compete with one another in various regionals ACM-ICPC competition in order to qualify for the prestigious ACM-ICPC World Finals, usually held in May.

During the competition, each team must solve as many problems as possible. They will submit their program for a specific problem throughout the contest. Each submission from teams will receive a verdict. The verdict is guaranteed to be one of the following:

No.	Verdict	Description
1.	AC	The submission is <u>accepted</u>, meaning that the program produces the same output as the judges' solution (AC = Accepted).
2.	TLE	The submission is <u>rejected</u> because the program exceeds the time limit given to solve the problem (TLE = Time Limit Exceeded). Note that the time limit here is applied to the runtime of the program, not the time elapsed since the beginning of the contest.
3.	MLE	The submission is <u>rejected</u> because the program uses more memory than the maximum memory allowed (MLE = Memory Limit Exceeded).
4.	WA	The submission is <u>rejected</u> because the program produces an output that is different from the judges' solution (WA = Wrong Answer).
5.	RTE	The submission is <u>rejected</u> due to a runtime error (RTE) during execution.

For every <u>accepted</u> submission, the penalty points of that team is increased by the time that the accepted submission is received along with the number of rejected submissions for that problem by that team, multiplied by 20 minutes.

For example, if a team solves problem "1" in the 40th minute, but has previously submitted two rejected submissions for "1", the team receives 40 + (2 * 20) = 80 minutes of penalty points.

A team receives the penalty points for a certain problem <u>if and only if</u> the team solves the problem. Rejected submissions for unsolved problems <u>will not count</u> towards the penalty points of a team until the team solves that particular problem.

The teams are ranked based on the number of accepted submissions first, followed by the penalty points. For teams that has solved the same number of problems, those with lower penalty points will be ranked higher.

In order to run this competition, they need a system to maintain the teams and all of the submissions. Therefore, the ACM has employed you to create the system for them. The system must be able to handle multiple queries defined in the input section below.

Good luck!

[1] The ACM-ICPC is a real contest. See https://icpc.baylor.edu/ for more details.

Input

The first line of input consists of three integers **N** (2 <= **N** <= 10), **P** (2 <= **P** <= 20), and **Q** (5 <= **Q** <= 500), each separated by a single space, representing the number of teams in the competition, the number of problems, and the number of queries being asked respectively.

The problems are named starting from "1" as the first problem, followed by "2", "3", and so on.

The next **N** lines will each contain the name of a specific team. It is guaranteed that all team names consist of English alphabets ('A'-Z' and 'a'-'z') and numbers ('0'-'9') only.

The next **Q** lines will contain a single query each with the following format:

Query Type	Input Format: `<QUERY_TYPE> <APPROPRIATE_PARAMETERS>`
1.	`SUBMIT TEAM_NAME PROBLEM TIME VERDICT` The team with the name `TEAM_NAME` submits their program for problem `PROBLEM` (1 <= `PROBLEM` <= `P`) at time `TIME` (1 <= `TIME` <= 10000) and receives the verdict `VERDICT`. The time given will be in minutes after the contest starts. It is guaranteed that the team exists and all `SUBMIT` queries will be given in order, i.e. a `SUBMIT` query with `TIME=A` will come before another `SUBMIT` query with `TIME=B` if and only if `A <= B`. It is also guaranteed that, at any single moment in the contest, a team can only submit one submission, i.e. if `A = B` (for the time), then the team must be different. If the team `TEAM_NAME` has submitted and solved this problem before, print "problem already solved", and ignore this submission. Otherwise, print the team name, the verdict, the problem name, and the number of previous submissions before this submission by the team, each separated by a single space.
2.	`DETAILS TEAM_NAME` Retrieves the team with the name `TEAM_NAME`. Print, in a single line, its name, number of accepted submissions, and total penalty points, separated by a single space. It is guaranteed that the team exists.
3.	`FIRST PROBLEM` Retrieves the team who solved the problem `PROBLEM` (1 <= `PROBLEM` <= `P`) first. For this query, print the name of the team and the time that the problem was solved, separated by a single space. If the problem `PROBLEM` has not been solved, print "problem `PROBLEM` has not been solved". It is guaranteed that the problem exists. If there are multiple teams who solved the problem at the same minute, print the team whose (accepted) SUBMIT query appeared first in the input.
4.	`UNSOLVED` Print, in a single line, all the problems that have not been solved in the contest, separated by a single space. There is no whitespace after the last problem name. If all the problems have been solved by at least one team, print "all problems have been solved".

5. TOP

   Print the details of the currently top-ranked team. This is equivalent of calling the "`DETAILS TEAM_NAME`" query, with `TEAM_NAME` being the name of the currently top-ranked team in the contest.

Output

Print the result of each query as described in the input format above. The last line of the output must contain a newline character.

Sample Input

```
3 2 11
TeamTam
ThanQ
HalimArmyPlatoon
UNSOLVED
SUBMIT ThanQ 2 2 AC
SUBMIT TeamTam 1 3 TLE
DETAILS TeamTam
SUBMIT TeamTam 1 4 AC
UNSOLVED
FIRST 1
FIRST 2
DETAILS ThanQ
TOP
DETAILS TeamTam
```

Sample Output

```
1 2
ThanQ AC 2 0
TeamTam TLE 1 0
TeamTam 0 0
TeamTam AC 1 1
all problems have been solved
TeamTam 4
ThanQ 2
ThanQ 1 2
ThanQ 1 2
TeamTam 1 24
```

Explanation

There are 3 teams and 2 problems in the contest. You are given 10 queries.

Initially, all the problems ("1" and "2") are unsolved. The team "ThanQ" submitted an accepted solution on the 2nd minute of the contest for problem "2". The team "TeamTam" submitted a rejected (TLE) solution on the 3rd minute of the contest for problem "1". After this, TeamTam's details are still unchanged since the penalty points is only applied after the problem is solved. TeamTam finally solved problem "1" on the 4th minute, earning 24 penalty points.

After this, all the problems have been solved. The first team to solve problem "1" is TeamTam on the 4th minute, and the first team to solve problem "2" is ThanQ on the 2nd minute. The top team is now ThanQ who have solved 1 problem with 2 penalty points, hence the query "TOP" and "DETAILS ThanQ" will now give the same output. The last query, "DETAILS TeamTam", shows that the team TeamTam has solved 1 problem with 24 penalty points.

Skeleton

You are given the skeleton file **ICPC.java.** You should see a file with the following content when you open it, otherwise you might be in the wrong directory.

```java
/**
 * Name         :
 * Matric No.   :
 * PLab Acct.   :
 */

import java.util.*;

public class ICPC {

    // define your own attributes, constructor, and methods here

    private void run() {

    }

    public static void main(String[] args) {
        ICPC competition = new ICPC();
        competition.run();
    }
}

class Problem {
    // define your own attributes, constructor, and methods here
}

class Team {
    // define your own attributes, constructor, and methods here
}
```

Notes:

1. You should develop your program in the subdirectory `ex1` and use the skeleton java file provided. You should not create a new file or rename the file provided.
2. You only need to modify the skeleton file. <u>You do not need to create a new file for each class</u>. All code should be <u>inside the file given to you</u> in the `ex1` directory.
3. If your algorithm is different from the given skeleton, you are free to write a solution according to your own algorithm. You are free to define your own classes besides the ones given in the skeleton file.
4. You <u>must (and need to)</u> use OOP for this sit-in lab.
5. You are free to define your own methods.
6. Please be reminded that the marking scheme is:

Input	: 10%
Output	: 10%
Correctness	: 50%
Programming Style	: 30%, which consists of:

   - Meaningful comments (pre- and post- conditions, comments inside the code): 10%
   - Modularity (incremental programming, proper modifiers [public / private]): 10%
   - Proper Indentation: 5%
   - Meaningful Identifiers (for both method and variable names): 5%

Compilation Error : Deduction of **50% of the total marks obtained**.