A. Football Betting

time limit per test: 60 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

In this problem, you have to implement a strategy for making bets on football (soccer) matches. Your program will need to interactively make bets on the game's result (whether the home team wins, the away team wins, or the game ends in a draw) and maximize profits.

For each match, the following parameters are known in advance: the division in which the teams play, the start time of the match, the identifiers of the home and away teams, the referee's identifier, and the bookmakers' coefficients on the result of the match.

After the match, the following parameters become known: the number of teams' goals in the first half and for the entire time of the match, the number of teams' shots (attempts), the number of teams' shots on target, the number of corner kicks, the number of yellow and red cards.

Describing data for training and testing

To train your strategy, you can use the training dataset available by the link http://assets.codeforces.com/rounds/1522/3056cab52876d3327edafe4771c8589a96cb1a93/train.csv.

Please note that using historical and any other third-party datasets for solving the problem **is prohibited**.

The training data is provided as a CSV file. The first line contains the fields' names:

- `Division` — the division in which the teams play (within which the match was played);
- `Time` — the time in the `HH:MM` format in which the teams play, excluding the time zone (ie local time at the match venue). The field may be empty;
- `home_team` — number identifier of the home team;
- `away_team` — number identifier of the away team; Not equal to the field owner team identifier;
- `full_time_home_goals` — the number of goals scored by the home team during the entire match;
- `full_time_away_goals` — the number of goals scored by the away team during the entire match;
- `half_time_home_goals` — the number of goals scored by the home team in the first half;
- `half_time_away_goals` — the number of goals scored by the away team in the first half;
- `Referee` — referee's identifier number. If there is no information, the field contains -1;
- `home_shots` — number of shots (attempts) by the home team;
- `away_shots` — number of shots (attempts) by the away team;
- `home_shots_on_target` — number of shots on target of the home team;
- `away_shots_on_target` — number of shots on target of the away team;
- `home_fouls` — number of the home team fouls;
- `away_fouls` — number of the away team fouls;
- `home_corners` — number of the home team corner kicks;
- `away_corners` — number of the away team corner kicks;
- `home_yellow_cards` — number of yellow cards for the home team;
- `away_yellow_cards` — number of yellow cards for the away team;
- `home_red_cards` — number of red cards for the home team;
- `away_red_cards` — number of yellow cards for the away team;
- `home_coef` — home team winning coefficient;
- `draw_coef` — draw coefficient;
- `away_coef` — away team winning coefficient.

Any of the fields `home_shots`, `away_shots`, `home_shots_on_target`, `away_shots_on_target`, `home_fouls`, `away_fouls`, `home_corners`, `away_corners`, `home_yellow_cards`, `away_yellow_cards`, `home_red_cards` and `away_red_cards` may be absent.

Input

The first line contains a single integer n — the number of matches, $1 \leq n \leq 8500$. This is followed by n matches in an interactive format. Matches are given in chronological order.

Interaction

Each match will be given to your program in the following form:

- On a separate line, 8 numbers separated by a space are given: values of the fields `Division`, `Time`, `home_team`, `away_team`, `Referee`, `home_coef`, `draw_coef`, and `away_coef` respectively, that is, the characteristics known before the start of the match. Fields `Referee` and `Time` may be absent, then -1 will be passed instead.

- In response to this, your program must print one of four lines: "`HOME`", "`DRAW`", "`AWAY`" or "`SKIP`". "`HOME`" — home team bet, "`DRAW`" — draw bet, "`AWAY`" — away team bet, "`SKIP`" — don't make a bet. After that, your program should print the line feed and flush the output buffer.
- After that, post-match characteristics will be given to your program. On a separate line, 16 numbers separated by a space are given: values of the fields `full_time_home_goals`, `full_time_away_goals`, `half_time_home_goals`, `half_time_away_goals`, `home_shots`, `away_shots`, `home_shots_on_target`, `away_shots_on_target`, `home_fouls`, `away_fouls`, `home_corners`, `away_corners`, `home_yellow_cards`, `away_yellow_cards`, `home_red_cards` and `away_red_cards` respectively. Any of the fields `home_shots`, `away_shots`, `home_shots_on_target`, `away_shots_on_target`, `home_fouls`, `away_fouls`, `home_corners`, `away_corners`, `home_yellow_cards`, `away_yellow_cards`, `home_red_cards` and `away_red_cards` may be absent, in which case -1 will be passed instead of the value.

After every response, your program should print a line feed and flush the output buffer. Otherwise, you will get the verdict `The solution is "hung"`. You can use `stdout.flush()` or `print(..., flush=True)` in Python to flush the output buffer. For other languages see the corresponding documentation.

Scoring

Points for one match are counted as follows: if your bet loses (that is, the prediction is wrong), you get -1. If you skip a bet, you receive 0 points. Otherwise, your bet is won and you get $\alpha - 1$, where $\alpha =$`home_coef`, `draw_coef`, or `away_coef` depending on the result of the match (home team win, draw, or away team win).

The points for submission are equal to the sum of all points for the matches.

The **last successful** submission of each participant will be used in the standings, that is, the latest submission with the verdict "`Partial`".

After the end of the contest, the last successful submissions of each participant will be re-tested on a new dataset. Only the results of these submissions on the new tests will be taken into account.

Please note that if according to the results of the final retesting you take a prize, you must be ready to provide the jury with all the supporting code, including the code for training the model, and, possibly, its description. In case of refusal or impossibility of providing, you will be considered as participating out of the competition.

Also, keep in mind that only preliminary tests are performed during the contest and your solutions will **not be tested** at the maximum possible n. Therefore, we recommend that you have a margin of execution time 3-4 times to be sure that a solution will meet the time limit during the final system testing.

Example

input

```
3
0 -1 152 426 1 3.627689584601 3.577721408504 2.198599748243

0 0 0 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
1 -1 141 217 -1 1.714712084927 4.034616670416 5.587944088527

3 0 1 0 17 6 11 2 11 16 10 3 0 1 0 0
17 19:00 333 444 7 3.242665536417 3.714271486813 2.242156197892

2 4 1 0 7 12 4 8 11 -1 -1 -1 -1 -1 -1 0
```

output

```


DRAW


HOME


AWAY
```

Note

The test from the statement is intended only to illustrate the interaction protocol, it differs from the first test.

You can use the additional language "Python 3 ZIP + libs" to solve this problem. Submission in this language should be a zip archive with the files of your solution. The archive must contain the file `main.py` at the root level — the entry point of your solution, it will be launched by the interpreter. The archive may contain directories and any auxiliary files that will be available for reading at runtime.

The maximum size of the archive file is 8 megabytes, unpacked is 32 megabytes.

Also, the followed additional libraries are available in this Python installation (3.8.6, 64 bits):

- cachetools (4.2.2)
- chardet (4.0.0)
- catboost (0.25.1)
- cycler (0.10.0)
- flatbuffers (1.12)
- future (0.18.2)

- graphviz (0.16)
- h5py (2.10.0)
- joblib (0.17.0)
- Keras (2.4.3)
- Keras-Preprocessing (1.1.2)
- kiwisolver (1.3.1)
- lightgbm (3.2.1.99)
- matplotlib (3.4.2)
- numpy (1.19.2)
- opt-einsum (3.3.0)
- pandas (1.1.2)
- patsy (0.5.1)
- Pillow (7.2.0)
- plotly (4.14.3)
- protobuf (3.17.0)
- pyparsing (2.4.7)
- python-dateutil (2.8.1)
- python-sat (0.1.6.dev9)
- pytz (2020.1)
- PyYAML (5.3.1)
- retrying (1.3.3)
- scikit-learn (0.23.2)
- scipy (1.5.2)
- six (1.15.0)
- sklearn (0.0)
- statsmodels (0.12.0)
- tensorflow (2.5.0)
- tensorflow-estimator (2.5.0)
- Theano (1.0.5)
- threadpoolctl (2.1.0)
- torch (1.6.0+cpu)
- torchvision (0.7.0+cpu)
- typing-extensions (3.7.4.3)
- wrapt (1.12.1)
- xgboost (1.4.1)