HackerRank | Prepare Certify Compete Apply Search 1

All Contests > SRBD Code Contest - 2024 (Round 2) > Punch-In/Out

Punch-In/Out

locked

Problem Submissions Discussions

Time Limit: $C/C{+}{+}\ (1s)$, $Java\ (2s)$
Memory Limit: $512MB$

In FRBD (*Fun Research & Development Institute Bangladesh*), employees follow a complex but fun office in-out system. An employee starts the office with the first punch-in through the system and ends the office with the last punch-out.

During the day, an employee can punch in or out an unlimited number of times. The employee is inside the office after a punch-in until the next punch-out. This time is measured as $\boldsymbol{punch\_out\_time} - \boldsymbol{punch\_in\_time}$. The total duration while the employee is inside the office is known as the *Working Hour* $-W$. Note that, there are some special times when W does not count even if the employee is inside that are described later.

The *Working Day* D is a metric that is measured based on W and some in-out system rules, then assigned to the employee on that day. Possible values of D are - *"Overtime"*, *"Full Day"*, *"1st Half"*, *"2nd Half"* and *"Absent"*.

Firstly, there are some hourly requirements for the values of D.

- Overtime: $W \geq 12$ hours.
- Full Day: $W \geq 8$ hours.
- 1st or 2nd Half: $W \geq 4$ hours. (2nd Half is only considered when 1st half is not applicable)

The punch system is as followed.

- Employees can punch in or out between $00:00$ to $23:59$.
- It's not possible to punch in and out simultaneously at the same minute.
- Punch out at $00:00$ is a special case, considered the last punch of the previous day. (see *sample* for more clarity)

The working hour W is calculated with these rules.

- W count starts from $07:00$, even if there are earlier punches.
- The first punch-in of the day after $11:00$ loses the *1st Half* (along with *Overtime* or *Full Day*): W does **not** start count until the *2nd Half* starts from $14:00$.
- The lunch hour is from $13:00$ to $14:00$ and W does **not** count during this time.

An employee will be considered *Absent* if any of the following is true:

- The first punch-in of the day is after $15:00$
- $W < 4$ hours or there is no punches throughout the day
- None of the other values of *Working Day* D is possible

Finally, the special *Late* case is applicable when the employee is not *Absent*:

- For *Full Day*, *1st Half* or *Overtime*: The first punch-in of the day after $10:30$ causes **Late**.
- For *2nd Half*: The first punch-in of the day after $14:30$ causes the **Late**.

Now, your task is to determine the *Working Day* D based on the in-out system rules mentioned above.

Input Format

Input starts with an integer T, denoting the number of test cases.

Each case contains a non-negative integer N, denoting the number of punch-in or out on a single day.

Each of the following N lines contains a space-separated special character '>' or '<', and a timestamp in $HH:MM$ format.

- '>' denotes the punch-in and '<' denotes the punch-out.
- The first input is punch-in and the last input is punch-out.
- Punch-in and punch-out inputs are given **alternatingly** and **chronologically**.

Please check the samples for a better understanding.

Constraints

$1 \leq T \leq 100$

$0 \leq N \leq 100$

$00 \leq HH \leq 23$

$00 \leq MM \leq 59$

Output Format

For each case, print the value of D.

Add a space-separated special character '~' at the end of D (except for "Absent"), in case of **Late**.

Please check the samples for a better understanding.

Sample Input 0

```
5
2
> 00:00
< 00:00
4
> 11:00
< 13:30
> 14:00
< 19:30
4
> 07:00
< 07:01
> 14:00
< 22:00
2
> 11:30
< 00:00
2
> 15:30
< 19:30
```

Sample Output 0

```
Overtime
1st Half ~
Full Day
2nd Half
Absent
```

Submissions: 26

Max Score: 100

Rate This Challenge:

☆ ☆ ☆ ☆ ☆

More

C

```c
#include <math.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <assert.h>
#include <limits.h>
#include <stdbool.h>

int main() {
    /* Enter your code here. Read input from STDIN. Print output to STDOUT */
    return 0;
}
```

Line: 1 Col: 1

Upload Code as File

□ Test against custom input

Run Code

Submit Code