Problem N. Note Offset Adjustment

时间限制: 2 seconds
内存限制: 512 megabytes

“双星很难吗？我 rks12.66。”

众所周知，Phigros 需要校准谱面延时。在遥远又可能不遥远的未来，克露丝卡尔酱发明一个机器能够自动玩 Phigros，但是她竟然忘记了机器设置的谱面延时是多少?

因此，她获得了全部 Bad 的好成绩，并生气地离开了实验室。你来到了实验室，看到了一片狼藉的机器。为了帮助克露丝卡尔酱，你决定偷偷地测量出机器的谱面延时?

这是一道交互题。

交互库有一个整数 $x \in [-400, 600]$，你每次可以询问一个整数 $y \in [-1000, 1000]$，然后交互库会根据以下情况返回答案：

- “Perfect”：$|x - y| \in [0, 80]$。
- “Good”：$|x - y| \in (80, 160]$。
- “Bad”：$|x - y| \in (160, 180]$。
- “Miss”：$|x - y| \in (180, \infty)$。

你需要在 10 次询问内返回正确的结果 x。

交互方式

本题有多组测试数据。

第一行，输入数据组数 $T(1 \leqslant T \leqslant 1000)$。

对于每组数据：

当你需要进行询问时：

- 以 “? y” 的格式输出并**清空缓冲区**。你需要保证 $-1000 \leqslant y \leqslant 1000$，且目前询问次数（包含该次询问）不超过 10 次，否则交互库会出现未知错误。
- 之后，请输入一个字符串 s 表示获取的字符串信息。

当你已经获得答案后，以 “! x” 的格式输出并**清空缓冲区**。

当处理完所有数据组数后，请立刻终止程序。

样例

standard input	standard output
1 Perfect Good Miss	? 0 ? 100 ? 200 ! 0

提示

当答案为 0 时，分别询问 $0, 100, 200$ 的结果如样例所示。

如何**清空缓冲区**：

- 在 C 和 C++ 中，使用 `fflush(stdout)`（如果你使用 `printf`）或 `cout.flush()`（如果你使用 `cout`）。
- 在 Python 中，使用 `stdout.flush()`。
- 特别地，在 C++ 中，使用 `cout<<endl` 会自动清空缓冲区。

如果你仍然对于交互题有疑问，下面会展示一份样例代码：

```
#include<stdio.h>
#include<string.h>
#include<iostream>
int main(){
    int t; cin>>t;
    while(t--){
        cout<<"? "<<0<<endl;
        string s;cin>>s;
        printf("0\n"); fflush(stdout);
    }
    return 0;
}
```

如果你无事可干，可以尝试如何在 9 次询问内解决该问题。