7553 Krypton Stadiums

The planet of Krypton contains n cities. These cities are located in distinct locations all along one straight line running west-east. The cities are labelled $0, 1, 2, \ldots, n-1$ in order from west to east. Each city is home to one team and one stadium. Each stadium will have two corresponding integers (a_i and b_i) defining an interval of cities that may play at that stadium. That is, a team from city x may only play at stadium i if $a_i \leq x \leq b_i$. It is guaranteed that every team will be able to play at its home stadium ($a_i \leq i \leq b_i$).

You have been hired to make the schedule for the upcoming season and must determine if the layout of the stadiums and cities is *great*, *acceptable* or *bad*.

- The layout is *great* if for every pair of cities, c_1 and c_2, there is a stadium in between c_1 and c_2 (inclusive) that can host the teams from both c_1 and c_2.
- The layout is *acceptable* if it is not *great*, but for every pair of cities, c_1 and c_2, there is some stadium that can host the teams from both c_1 and c_2.
- The layout is *bad* if there is some pair of cities where no stadium can host the teams from both cities.

Input

The input will contain multiple test cases.

The first line of each test case will contain an integer n ($2 \leq n \leq 200\,000$) denoting the number of cities. The next n lines will give the intervals of each stadium. The intervals are given by exactly 6 characters. The first three characters will denote a_i and the last three characters will denote b_i (whose definitions are given above). Each set of three characters will denote a base 62 number (using the ordering `0-9A-Za-z` as our alphabet). For example, cities 0, 1, 9, 10, 35, 36, 61, 62 and 199 999 are represented by '`000`', '`001`', '`009`', '`00A`', '`00Z`', '`00a`', '`00z`', '`010`' and '`q1n`', respectively.

Input will be terminated by end of file. There will be no more than 1 000 different test cases and there will be no more than 2 000 000 stadiums across all test cases.

Output

For each test case, output one of three strings: '`Great`', '`Acceptable`' or '`Bad`'.

Sample Input

```
4
000001
000003
002002
002003
4
000000
001001
002002
003003
4
```

```
000001
000003
002002
003003
```

Sample Output

```
Great
Bad
Acceptable
```