# Perfect

| | |
|---|---|
| Input file: | `standard input` |
| Output file: | `standard output` |
| Time limit: | 1 second |
| Memory limit: | 256 megabytes |

A classic problem in mathematics is to determine if a given positive integer is **perfect**. We call a positive integer, $N$, **perfect** if it is equal to the sum of its proper positive divisors, that is, the sum of its positive divisors excluding the number itself. (Note that a proper divisor of $N$ is a positive integer **less** than $N$ which evenly divides $N$. For example, the proper divisors of 6 are 1, 2 and 3, and the proper divisors of 8 are 1, 2 and 4).

Examples of **perfect** numbers include 6 and 28, noting that the proper divisors of 6 are 1, 2 and 3 and that $1 + 2 + 3 = 6$. The proper divisors of 28 are 1, 2, 4, 7 and 14, and also that $1 + 2 + 4 + 7 + 14 = 28$.

As the vast majority of numbers are not perfect, we can furthermore divide non-perfect numbers into two sets. We denote a positive integer $N$ as **abundant** if the sum of its proper divisors is **greater** than $N$, otherwise we denote $N$ as **deficient** if the sum of its proper divisors is **less** than $N$.

Note that 9 is a deficient number, since the proper divisors of 9 are 1 and 3, and $1 + 3 = 4$ which is **less** than 9. Also, 12 is an abundant number, since the proper divisors of 12 are 1, 2, 3, 4 and 6, and $1 + 2 + 3 + 4 + 6 = 16$ which is **greater** than 12.

Your task is to write a program that, given a single positive integer $N$, will determine if it is **perfect**, **deficient**, or **abundant**.

## Input

Input consists of a single integer $N$.

## Output

Output a single word, either `Abundant`, `Deficient`, or `Perfect`, if $N$ is abundant, deficient, or perfect respectively.

## Examples

| standard input | standard output |
|---|---|
| 12 | Abundant |
| 9 | Deficient |
| 28 | Perfect |