F. Level Generation

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Ivan is developing his own computer game. Now he tries to create some levels for his game. But firstly for each level he needs to draw a graph representing the structure of the level.

Ivan decided that there should be exactly n_i vertices in the graph representing level i, and the edges have to be bidirectional. When constructing the graph, Ivan is interested in special edges called *bridges*. An edge between two vertices u and v is called a *bridge* if this edge belongs to every path between u and v (and these vertices will belong to different connected components if we delete this edge). For each level Ivan wants to construct a graph where at least half of the edges are *bridges*. He also wants to maximize the number of edges in each constructed graph.

So the task Ivan gave you is: given q numbers $n_1, n_2, ..., n_q$, for each i tell the maximum number of edges in a graph with n_i vertices, if at least half of the edges are *bridges*. **Note that the graphs cannot contain multiple edges or self-loops**.

Input

The first line of input file contains a positive integer q ($1 \le q \le 100\,000$) — the number of graphs Ivan needs to construct.

Then q lines follow, i-th line contains one positive integer n_i ($1 \le n_i \le 2 \cdot 10^9$) — the number of vertices in i-th graph.

Note that in hacks you have to use $q = 1$.

Output

Output q numbers, i-th of them must be equal to the maximum number of edges in i-th graph.

Example

input
3 3 4 6
output
2 3 6

Note

In the first example it is possible to construct these graphs:

1. 1 - 2, 1 - 3;
2. 1 - 2, 1 - 3, 2 - 4;
3. 1 - 2, 1 - 3, 2 - 3, 1 - 4, 2 - 5, 3 - 6.