

[Advent of Code](#)
[\[About\]](#)
[\[AoC++\]](#)
[\[Events\]](#)
[\[Settings\]](#)
[\[Log Out\]](#)
 Harry Comp 6★
[0x0000|2017](#)
[\[Calendar\]](#)
[\[Leaderboard\]](#)
[\[Stats\]](#)
[\[Sponsors\]](#)

--- Day 3: Spiral Memory ---

You come across an experimental new kind of memory stored on an infinite two-dimensional grid.

Each square on the grid is allocated in a spiral pattern starting at a location marked `1` and then counting up while spiraling outward. For example, the first few squares are allocated like this:

17	16	15	14	13
18	5	4	3	12
19	6	1	2	11
20	7	8	9	10
21	22	23	---	...

While this is very space-efficient (no squares are skipped), requested data must be carried back to square `1` (the location of the only access port for this memory system) by programs that can only move up, down, left, or right. They always take the shortest path: the **Manhattan Distance** between the location of the data and square `1`.

For example:

- Data from square `1` is carried `0` steps, since it's at the access port.
- Data from square `12` is carried `3` steps, such as: down, left, left.
- Data from square `23` is carried only `2` steps: up twice.
- Data from square `1024` must be carried `31` steps.

How many steps are required to carry the data from the square identified in your puzzle input all the way to the access port?

Your puzzle answer was `480`.

--- Part Two ---

As a stress test on the system, the programs here clear the grid and then store the value `1` in square `1`. Then, in the same allocation order as shown above, they store the sum of the values in all adjacent squares, including diagonals.

So, the first few squares' values are chosen as follows:

- Square `1` starts with the value `1`.
- Square `2` has only one adjacent filled square (with value `1`), so it also stores `1`.
- Square `3` has both of the above squares as neighbors and stores the sum of their values, `2`.
- Square `4` has all three of the aforementioned squares as neighbors and stores the sum of their values, `4`.
- Square `5` only has the first and fourth squares as neighbors, so it gets the value `5`.

Once a square is written, its value does not change. Therefore, the first few squares would receive the following values:

147	142	133	122	59
304	5	4	2	57
330	10	1	1	54
351	11	23	25	26
362	747	806	---	...

Our [sponsors](#) help make Advent of Code possible:

[SmartyStreets](#) -
[U2VuZGluZyBDaH](#)
[Jpc3RtYXMGY2Fy](#)
[ZHMgdG8gYmFkIG](#)
[FkZHJlc3Nlc38K](#)

What is the first value written that is larger than your puzzle input?

Your puzzle answer was .

Both parts of this puzzle are complete! They provide two gold stars: **

At this point, you should [return to your advent calendar](#) and try another puzzle.

Your puzzle input was .

You can also [\[Share\]](#) this puzzle.