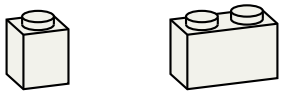
Short Take-Home Exam 2

(Deadline: 28th of April 2023, Friday, 23:59)

1 Problem

You are given two kinds of toy bricks: (a) *single* bricks of size $1 \times 1 \times 1$, and (b) *double* bricks of size $2 \times 1 \times 1$ (where the dimensions are width, height and depth). They are shown below:

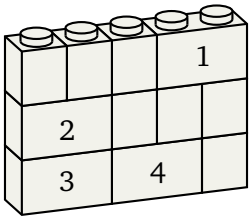


Toy bricks are always used in upright position. The faces of the bricks are made from identical material and are indistinguishable. As famously known, the bricks have hollow tubes on their bottom and studs on their top. Two bricks can be locked through their tubes/studs, by placing one brick on top of the other (such that at least one tube/stud pair is coincident).

We want to construct a thin upright wall of width W and height H using these bricks. Such a wall always has depth 1. The wall is fully filled with bricks; it contains no holes in its interior or on its boundary. *The wall may be connected or not.* In other words, the wall can consist of sections which are not (directly or indirectly) locked to each other.

Unfortunately, the supply given to you is very precise. There are D double bricks and $(WH - 2D)$ single bricks. Consequently, a wall of size $W \times H$ uses all of the brick supply and contains *exactly* D double bricks.

The following is *an example of a valid wall* for the parameters $W = 5$, $H = 3$ and $D = 4$:



Let us iterate again that a valid wall may be connected or disconnected (as above).

In how many different ways can we construct such a wall? Write a program that answers this question given W, H and D. Since the number of ways can be quite large, your program is supposed to answer this question *modulo 8191*.

The front face of the wall is significant while counting. A wall and its mirror image (equivalently, its 180° rotation) are distinct configurations, unless the wall structure is symmetric along its width.

2 Input Format

```
W H D
```

`W` = Width of the wall. `H` = Height of the wall. `D` = The number of double bricks.

3 Output Format

```
N
```

`N` = The number of ways to construct such a wall modulo 8191.

4 Limits

- $1 \leq W, H \leq 150$
- $0 \leq D \leq \frac{WH}{2}$
- Time Limit: 1 second, Memory Limit: 256 MB, Stack Limit: 32 MB

5 Clarifications

- Your solution is expected as a C++ program source named `sthe2.cpp` that reads from the standard input and writes to the standard output.
- It is OK to copy code from the sample codes we shared in ODTÜClass or from your individual previous THE submissions. Copying from elsewhere will be considered cheating.
- You are supposed to submit your code via the VPL item in ODTÜClass. Use the "evaluate" feature to run the auto-grader on your submission.
- The grade from the auto-grader is not final. We can do further evaluations and adjust your grade. Submissions that do not attempt a "reasonable solution" to the given task may lose points.
- Your code will be compiled with the options: `-std=c++2a -O3 -lm -Wall -Wextra -Wpedantic`
- Late submissions are not allowed.

6 Sample Input/Output Pairs

Input	Output
2 5 4	5
3 5 4	80
4 4 4	195
5 2 4	9

Input	Output
5 2 5	0
6 6 1	30
6 6 2	411
6 6 9	2714

Input	Output
6 6 18	1
10 10 25	5667
20 20 100	810
100 100 2500	6088

7 Hints

- We suggest using dynamic programming (ideally in bottom-up form) to solve this problem.
- This problem is similar to your LTHE2 but involves an additional parameter. This parameter should probably play a role in your recurrence relations.
- We emphasize that the walls need not be connected; this simplifies the problem in some ways.
- Please, be careful when writing out the base cases.
- Pay attention to memory/stack limits. You may lose points if you exceed either, despite your algorithm being sufficiently fast.
- We repeat our warnings about the modulo operation (as done in the previous THE): Apply it before getting an overflow/underflow and do not apply it on negative values.
- There is an $O(WHD)$-time algorithm that can get full marks on this THE (if implemented well).
- There also exists an $O(WH^2D)$-time algorithm which can get 80% of the points.
- A well-implemented exhaustive search with some basic pruning can earn 40% of the points.
- If you are stuck, we suggest that you relate this problem to LTHE2. Try to adapt/extend the two algorithms that we studied for LTHE2. (The ones with the running times $O(W^2 + W\log H)$ and $O(WH^4)$.) Thinking about an exhaustive search may also help to find some algorithms.