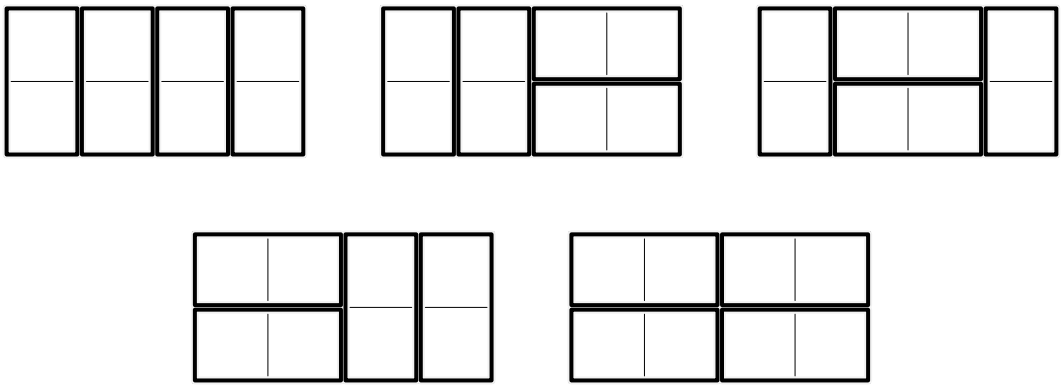
Different Dominoes Divisions

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	1024 mebibytes

Ivan has a checkered rectangle of size $2^k \times 2^{k+1}$ and wants to split it into domino pieces (rectangles of size 1×2 or 2×1).

To do that, he developed the following recursive algorithm:

- If $k = 0$, then the rectangle has size 1×2, and splitting is done.
- Otherwise, consider our rectangle as a rectangle of size 2×4 consisting of big cells of size $2^{k-1} \times 2^{k-1}$. There are 5 possible ways to split it into domino pieces (see below). Select some way of splitting. The resulting pieces will be of size $2^{k-1} \times 2^k$ or $2^k \times 2^{k-1}$. Split each of them recursively.



Ivan likes his generator a lot. It can generate many non-trivial domino tilings. He generated some tiling and left the file with the tiling on the computer. Unfortunately, the computer was broken and information about some domino pieces in the tiling was lost.

You are given some non-intersecting domino pieces in the rectangle of size $2^k \times 2^{k+1}$. Information about cells not covered with domino pieces is lost. Find the number of different tilings that can be generated by the algorithm above such that all given domino pieces are present in the tiling.

More formally, a tiling can be generated by the algorithm if there exists a way to split the rectangle of size 2×4 on each step such that the resulting tiling is as needed. Two tilings are considered different if the sets of domino pieces in them are different.

Since the number of different tilings can be large, count them modulo 998 244 353.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^5$). The descriptions of the test cases follow.

The first line of each test case contains a single integer k ($0 \le k \le 10$).

The next 2^k lines describe the tiling of the board, row by row from top to bottom. Each of these lines contains 2^{k+1} characters, describing the cells in the corresponding row from left to right. Each character is one of '`U`', '`D`', '`L`', '`R`', or '`.`', meaning that the cell is covered with a top, bottom, left, right half of a domino or nothing, respectively.

It is guaranteed that the sum of 4^k over all test cases does not exceed 2^{20}.

Output

For each test case print a single integer — the number of different tilings that can be generated by the

algorithm modulo 998 244 353.

Example

standard input	*standard output*
5 0 .. 1 ...U ...D 2 .UU.LRUU .DD...DD .ULR...U .D..LR.D 2 .LR..... U....... D....... .... 3 LRLR.....ULR.... .U.......D..LR.. .D.......ULR.... LRLR.....D...... ....LRLR....LRLR ....U.......U... ....D.......D... ....LRLR....LRLR	1 3 2 0 7657344

Note

One possible tiling for the third test case is depicted below.