MODULO

An array of 42 boolean values (all initially false) allows us to keep track of which remainders had occurred in the input.

HERMAN

A circle in taxicab geometry is a square in regular geometry (at an angle of 45 degrees if when placed in a Cartesian coordinate system). The area of such a "circle" is thus $(\text{sqrt}(2)\cdot r)^2 = 2\cdot r^2$.

The area of a regular circle is $r^2\pi$. π can be embedded in the source code as a constant or calculated (better) as 4·atan(1) or acos(-1). Some implementations of the C language define the constant M_PI in the header <math.h>, though this is not defined by the C standard.

OKVIRI

There are two main approaches to implementing the solution; one is to try and produce output directly, the other (arguably easier to grasp) using a helper matrix of characters which is then output verbatim.

SLIKAR

The constraints in the task were low enough that a minute-by-minute simulation was efficient enough, keeping track of flooded fields and those that the painter could have reached by then.

A more efficient solution first uses breadth-first search to find for all fields the time at which they will become flooded. Another BFS then simulates the painter's movement, taking into account information gathered by the first BFS.

BOND

There are N! (N factorial) different ways of assigning missions to the agents. For N=20, this number is too large to go through all of them and find the best.

Observe that, if we have already assigned agents 1 through X to some X missions, then the probability of the remaining missions being successfully completed does not depend on exactly which of the X agents were assigned to the X missions. This fact allows us to use dynamic programming to solve the task.

The state in the search will be the subset of missions that have been assigned (implemented using a bitmask), which also contains the number of missions assigned so far (X). The space complexity is $O(2^N)$ and the time complexity is $O(N\cdot 2^N)$, which fits into the 1 second time and 32 MB memory limits.

DEBUG

To solve the problem efficiently, observe that, if square A is a killer, then every square inside it (sharing the same center point) is also a killer. Conversely, if a square is not a killer then no square surrounding it can be a killer.

This forms the basis of a $O(N^4)$ solution: try and extend a square from each memory cell. The worst-case complexity is $O(N^4)$ because there are N^2 cells, O(N) dimensions to be checked and checking the frame (when trying to extend a square killer) takes O(N).

Naively implementing the above approach will not be efficient enough. One way of speeding it up is to, for each cell and each of the 4 directions, precalculate 32 or 64 binary digits starting at the cell (y,x) in the chosen direction. This allows us to check frames in blocks of 32 or 64 bits (instead of one by one) and speed up the solution by a factor.

Other heuristics that reduce the number of checks (eliminating those that surely can't contribute to the solution) could be used to speed up the solution further.