1）Consider the number of possible configurations as below:

The number of grids in the array except for the last row is (R-1)C; for these cells, there are four possibilities each, with number 0, 1, 2, or 3, so the number of configurations in this part is $4^{(R-1}C)$.

The monkey can stand at one of the column, and may have life 1, 2, or 3. So the total configuration for the game is …

2）We prove the argument as follows: After 3R moves, either all balloons are already cleared (all cells have value 0), or the game is already dead.

If all balloons are already cleared, then the player wins the game and the moves are smaller than 3R. Consider a cell has nonzero value: since the balloon moves one cell (descent or re-spawn) every time the player makes a move, so the remaining balloon in the nonzero cell must have moved at least 3R cells. Given the number of rows = R, the balloon must have hit the ground three times at least, so the monkey has already lost 3 lives, then the player must have already lost the game.

So the maximum number of moves in a game is 3R

3) Design a directed graph. Every node is a configuration and directed edge from node 1 to node 2 exists when configuration 1 can lead to configuration 2 after one turn (while the monkey move left, right, or through a dart straight up.

Initialize the graph as with a node of the current configuration, then compute its neighbors, and neighbors of neighbors. Keep track of the “distance” of a node with the root node. When a node implies lost game, stop finding its neighbors. Find the closest neighbor that wins the game (can be easily done by a BFS), return its distance from the root or return None if no such node exists.

The number of configurations going through is bounded by $3^{3R}$ in 5(2) since the maximum number of moves is 3R which is the depth of the graph, and every node can have as many as 3 edges to the next level. To build a node takes O(RC) given the side of the board B. As a result the total runtime of this algorithm is $O(RC\*E^{3R})$. Also, to further decrease the average runtime, we can keep track of nodes visited and end finding adjacent nodes of a node that represented a lost game.

The correctness is proven by the nature of this algorithm: we explore all the possible decisions that the monkey can make. Since we can find all ways to win the game, we can surely find the minimum number of moves needed, or return None when no such way exists.