6.

a) Use Dijkstra's, with a graph only directed Right and down with end node being coal, start node being where the robot starts

b) Use Dijkstra's Again / BFS

c) Use Dynamic Programming

D) This is DFS,

✓

grid = 2d array of nodes that are spaces
end = grid[R-1][C-1]

recurse (position in grid):
    if (position = end):
        return 1
    if can go down and right:
        return recurse(Position[j-1]) + recurse(position[i+1])

    if only can go down
        return recurse(position[j-1])

    if only can go right
        return recurse(position[i+1])

Time complexity: O(V+E), this is because V is number of vertexes (possible squares) E is paths from those squares. Each path has at most 2 possible paths, it is a simple DFS algorithm

Space complexity: O(R·C), this is for the initial grid, nothing else is taking up space besides O(1) returns

One thing I could improve is by using some Dynamic programming algorithms, it might improve the time complexity a bit.