a) Briefly explain the time and memory complexity for both versions of your game. You can write your answer in a separate file and submit it together with the other programming submissions.

For both versions, if we consider the input size $n = d$, it seems to suggest that both version operate with time complexity of $O(n^2)$. This would make sense for recursion considering that although each recursive method could potentially call multiple other recursions (up to 4), all squares that were previously visited would return false immediately. The worst-case scenario would be traversing through each square of the $d\ x\ d$ board, so it would do recursion for around $d\ x\ d$ times. Hence, time complexity for the recursion is $O(n^2)$. The same thought process could apply for the iterative version, each loop only pushes a new position into the stacks if that position has not been visited yet. The worst-case scenario is once again to pass through each square of the board, thus the while loop will perform around $d\ x\ d$ times. Hence, time complexity for the iteration is also $O(n^2)$.

For both versions, space complexity seems to be $O(n^2)$. Since the algorithms is using a 2D array board, it is allocating $n\ x\ n$ memory, which is n^2. The algorithms also use another 2D array for the positions that were already visited, also allocating $n\ x\ n = n^2$ memory. In recursion, the maximum number of stack frames also reflect n^2, as there are about at most $n^2 + 1$ maximum frames (levels of recursions), since the algorithm at worst-case traverses and thus, allocates space for each square on the board. Adding the space allocated, a space complexity of $O(n^2)$ is obtained for the recursive method. In the iterative method, the algorithm creates two stacks whose maximum number of frames is at most $n\ x\ n = n^2$ since the algorithm at worst-case traverses and thus, allocates space for each square on the board. The stacks hold the possible row and column positions, respectively. Thus, adding all the space allocations together, a space complexity of $O(n^2)$ is obtained.

b) For the first version of your solution describe the type of recursion used in your implementation. Does the particular type of recursion have an impact on the time and memory complexity? Justify your answer. Also explain whether a tail-recursive version is possible. If yes, you can earn bonus marks by submitting such a version.

The type of recursion resembles multiple recursions. The type of recursion does have an impact on time and memory complexity. Multiple recursions create branching forks of recursions which use up time and memory resources through the number of recursions and using up stack frames. A tail-recursive version could help in saving said resources. This can be done by implementing a linear type of recursion for this algorithm.

c) For the second part of your solution, justify why you choose that particular data structure (e.g. why you choose a stack and not a queue, etc.). Explain whether your chosen data structures have an impact on the time and memory complexity.

Recursion uses the data structure of a stack. Thus, implementing an iterative version of the MagicBoard can also be done using a stack. That is why a stack was chosen over the other data structures such as a queue. This could have an impact on the time and memory complexity. Because the algorithm implements iteration using stacks and operates similarly to that of recursion, the time and space complexities become very similar to that of the recursive version.

d) For each version provide test logs for at least 20 different game configurations. They must be sufficiently complete to show that your solution works for various board dimensions and square values.

1) Recursion start position 0, 0

```
 4  0  3  4  4
 4  4  4  4  2
 4  4  2  3  1
 4  1  3  3  1
 4  3  3  3  1
false
```

2) Recursion start position 0, 0

```
 2  0  4  3  3
 3  1  2  4  3
 1  2  4  3  3
 2  1  2  3  3
 2  1  4  3  1
There is a solution!
true
```

3) Recursion start position 0, 0

```
 3  3  4  4  4
 2  4  1  3  1
 4  0  2  2  2
 3  4  2  3  4
 2  3  2  2  3
There is a solution!
true
```

4) Recursion start position 0, 0

```
 4  2  0  2  2
 3  3  2  1  1
 2  4  4  4  3
 2  3  4  1  2
 3  2  1  2  1
There is a solution!
true
```

5) Recursion start position 0, 0

```
 1  1  1  3  1
 4  4  4  1  0
 3  3  1  3  4
 4  1  3  2  3
 3  4  1  1  1
There is a solution!
true
```

6) Recursion start position 0, d-1

```
 3  3  4  0  4  3
 2  3  2  4  1  1
 5  4  4  1  2  5
 1  3  1  4  5  3
 4  5  4  3  1  4
 1  1  3  4  3  4
There is a solution!
true
```

7) Recursion start position 0, d-1

```
 3  6  0  2  4  3  5
 5  5  4  1  5  4  5
 5  1  4  1  4  6  6
 5  2  3  6  1  4  4
 2  1  1  1  6  3  1
 5  6  1  2  1  5  6
 3  3  6  6  1  3  4
There is a solution!
true
```

8) Recursion start position 0, d-1

```
 7  5  4  1  3  2  6  5
 4  1  2  3  3  3  0  3
 3  6  1  7  6  6  1  2
 2  6  5  1  2  4  3  1
 2  7  2  4  2  7  5  6
 7  3  6  1  3  7  3  5
 3  1  1  7  5  3  6  7
 3  3  3  2  7  6  5  6
There is a solution!
true
```

9) Recursion start position 0, d-1

```
 8  2  8  4  6  0  4  7  8
 5  5  4  1  1  5  6  4  8
 7  6  8  7  2  7  6  2  2
 7  4  3  2  4  1  6  5  4
 8  2  1  4  7  3  2  3  6
 3  5  7  5  8  8  5  1  3
 8  8  5  8  3  5  1  6  7
 8  1  4  1  8  4  2  1  3
 8  1  2  5  5  8  8  6  2
false
```

10) Recursion start position 0, d-1

```
 1  9  4  7  5  0  2  3  5  8
 4  8  2  5  2  1  5  2  8  2
 9  7  8  7  1  4  8  1  2  4
 8  7  3  9  7  1  1  1  2  8
 8  3  9  4  7  9  6  2  8  5
 5  6  2  7  8  3  3  3  4  9
 4  7  3  1  6  6  8  5  7  8
 3  5  7  3  4  3  1  2  1  1
 4  8  1  3  1  3  5  2  3  8
 5  2  3  7  7  4  2  6  2  6
There is a solution!
true
```

11) Recursion start position d-1, 0

```
 9  2  4  5  0  2  2  2  1 10  1
 4  7  5  7  1  9  2  4  6  3  8
 4  7  2  8  4  7  8 10  1  1  7
 3  9  3  1  9  1  5  9 10 10  9
10 10  7  5  4  1  8  1  2 10  7
 6 10 10  8  5  7  8  2  6  4  5
 1  8  3  8 10  1  4  4  2  4  9
 4  9  1  7  6  1  7  2  8 10  7
 3  7  1  8  9  9  4  7  8  7  3
 1  9  5  4  9  6 10  4  3  5  9
 4  4  1  2  9  4 10  4  9  1  4
There is a solution!
true
```

12) Recursion start position d-1, 0

```
 11 10  7  2  1  6  8  0  8  7  2  8
 11  5  4  4  6  4  8  5 11  5  7  2
  6 10  1 11  6  3  9  6  8  3  2 10
  5  3  2  9  3 10  8  9  8 10 11 11
  7  6  5  3  4  4 11  4  2  2 11  2
  9 11  1  3  2  8  9  7  8 10  1  7
  5  6  8  7 11 11  3  7 10 10 10  1
 10 11  6  7 10  6  9  4  4  3  6  2
 11  6  9  8  6 10  1  1 11  7  7 11
  5  5  5  2  2  4  2  5 11  7  3  6
  2  7  6  6  7 10  9  7  4  1 10  6
 11  6  4 10 10  8  7  9 10  2  8  9
There is a solution!
true
```

13) Recursion start position d-1, 0

```
  9  3  9  7  6  7 11 10  3 10  8  5  0
 12  8  1  8  2  9  5  5  9  7  5  7  9
  7  5  1  6  1 11  4  8 10  9  5  3  8
 12  3 11 11  6  4  5  6  6 11  5 12  1
  3 11  9  8  3 10 12  5  1 11  4 12 11
 11  9  7 11 11  9  6  8  6  9 10  2 12
  7  6  2  5  1  6  5  9  9  9  9  2 10
  9  7  4 10  3 11 10  8  3  8 11  6 10
  7  3  5  3  3  1  7  3 11 11  8 12  7
 12  3  6  4  7  9 12  6  2  5  5 10  6
  5  3  8  1  5  4  1 10  5  5  3  6  4
  9  8 11  2  1  8  9 10  5 12 12  7 10
 11  7  7 10 12  4  5  1  3 11  2 11 12
There is a solution!
true
```

14) Recursion start position d-1, 0

```
 12  5  4  1 12  4  4  4  2  7  5  4  8 12
  6  7  3  0 13  8  7 11  9  6  1  8  2  1
  9  7 12  2  8 11 11  6  8 11  8  9  5  4
  3 13 13 13  9 10  8  4  4  3  9  3  5 13
  1  3  4  8  3 10 12  9  9  4 10  9  9  8
  4  3  5  5  7  7  9 11  4  7  2  3  7 10
  2 13 13 11 12  9  3 12 12  6 13  4 11  6
  1 13  4  7  3  9 11 10  9  4  7  1  3  4
  8 11  4  3  4  5  5  5  8  1  9  6 13 12
  3  9  2  5  7  7  2  5 12  4  1  3  7  1
  9  1  2  9  9  3  8  9 11  1 11  5  6 12
  4  1 12  2  2  2  2  1  5 11  6  2  6 10
  3  3  7 10 12  5 11  8  9  9 11 11 13  8
  8 11 10 10  8  8  8  1 12  2  1  6  1  5
There is a solution!
true
```

15) Recursion start position d-1, 0

```
 11  1  2 10  6  9  5  1  2  7  9  0  6  6  9
 14  7 10  5  4  3  9  5  2  1 13 12 14 10 11
 11 11  3 12  5  8  1 10  1  8 10  8 12 12  3
  5  1 10 13 14  3 10  9  1  7 10  5  1  8  5
 13 11 13  9  2  3 12  9  9  7  7 13 14  6  6
  1  1  4  8  7  2  2 10 10  5 10 12  7  1  4
  1  1  6 11  8  2  4  9  2  8 10 13 12  3 11
 14  9  2  9  1  5  3  4 13  7  5 10  6  4 12
 14 14  4 14 11  8  5  2  9 12  1  8  4  9  4
  5 11  3 14  3 10 14  6  3  6 14 11  9  5  9
 12  2 10 12  6 11  1 11  8 13  2 12 13 12  7
 10  3  5  1  7  5 14 11  2  8  1  2  4 14 10
  1 10  1  9 12 10  7 13 13  9  7 12  8  6 14
  5 10 14 12 12  9  1 11 14 14  5  8 11 10  3
  3 13  8 10 13 11  2 13  1 10 13  6  4  6  4
There is a solution!
true
```

16) Recursion start position d-1, d-1

```
 15  6  7  9  4  5  7 12 15 12  8 14  9  2  4  9
  8 12 11  9  2 11  9  6  3 15 12  6  5 13 10 11
 15 15 12  4  6 11  4 15  1  8 13 12  0 12  9  3
  7  6 12  2 12  5  2 13  1 15  2  4  9 11 10 10
 12 11  5 10  3 10  8  9  1  9 13 15 10  6  1  7
 15  1  5 11  8  2 10  4 15  2  3  9  9 13 11 10
  8 12 14  6  8  4  6 11  2  8 12  9  5  3  4 12
  1  7 11 10  4 15 13  6  4  6  5 12  1 15 11  1
 10  2  2 15 10 15 11 10 14 12 12  5  1  5 10  4
 10  8  4 11  5 13 11  3 10  2  8  6 10  2  4  2
 13  1  8  8  3  6 10  5  2  8 12  3 14  9  7  6
 14 13 14  9 15 10  6  2  8  5  6  4 14 12 15  4
  8 12  7  4  9  7  7  2  4  1 14 13 11  1  7 10
  7 15 12 13 15  4  3 13  5  8  2  9  9  6  8 13
  3 13  2  2  7  2  4  9  1 11  3  4 11  7  6 15
  1 15 13  1  9  4  1 12 13 14  5 14  5  3 15 10
There is a solution!
true
```

17) Recursion start position d-1, d-1

```
 15  5  9  5 11 15 16  6 12  3  3  3  0 15 15  8 10
  2 14 10 11  1  2  4  8  4  2 10  6 16  8 11 13  3
  5  3 12  3 10  3 13  9  4  9 16  5  4  3 11 15  7
  3 16 15  1 15  5 13  5  4 13 16 16  5  4  5  9  6
  5 12  4  9 10 13  7 13  5 13  8 16  3  4 11  5 14
  6 13 14  7 13 14  3 15 15  9  7  9 15  3 11  5  2
  1 16  2 11  7  4  4  9 16  8  7 12  2  8 11 14  1
  1 15  9  9 13 15  7 11 14 12 11 12 16  1  6  2  2
  7  1 15  7 12  9 12 11 11  2  6 15  2 15  6  3 13
  8 16 11 11 11  9  3 12  3  6  9 11  9  3  4  1 11
 13 10 11  7 10 12 16 11  3  7  2  7  7  3  5  9  6
 14  5  7  2 10  2  2  8 15 12  3 11  4 15 14  7  4
  5  8  4  1  9  7  2  7 16  5 14 11  9  7 15  2 10
  6  2  4 15  2  8 12 14  5 15  4 12  8  6  6  8 13
 14  3  6  7 15  4 11  5 10 15  4 13 11 11  1  4 10
 13  5  5  4 10  6  2  6  1  8  6 13  8 10  9  6  8
  7  7  8 11  1  9  3 14 16 11  1  8  2  5 15 10 13
false
```

18) Recursion start position d-1, d-1

```
15 15  1  3 10 17  7 17  6  0  1 16  7  6  2  8 12 15
 1  1 17 12  7  5 16 17  9 15  4 17 10  5 15  8 17  3
 9  5 16 12  2  2  4  7 17  6 10 13 11  7 10 13  9 11
 7 16  3 15 15  9  3  7  5  7 11  1  8 12  1 11  3  5
 9 13 11  1  2 13 13  6 16  6  9 14  7 15  8 13  5 11
15  7  3  5 14  3 13  3  8 14  1  5 17 11  6 16  3 16
14  1  2  5 12 12 11 12 16 16  7  9  3 17 15  8 14 16
13  4 16 15 14  6 12  9  3  4  6 10 14  4  5 13  8 15
 2  5  4 13  5 10  4  2  5 14 14 13 11  3 10 10 13  4
 2  8 11 12 17 15  2 16 13  5  1  7  7  7 10  5 15 15
14  2  9  6  9  6 17 12 11  9 11  5 15 13  3 13 16 15
 8  6  8  8  1 12  2  2  8 15 17  9 16 12 16 12  5  3
 7  8  4  9 10 15 16 16 13  3 11  5 17  4  5 11  9 13
10  8  7  5  5  2  1 10  1 14 14  1  6 13 11 13 12 14
 7  4 17  8 14 15  4  4 12  6  6  1 12 10  1  4  2 13
13  5  6  5 10  5  8  7  3  5 14  1 17 15 16 11 13  8
 1  6 15 16  3  5 11  8 15  4  6 16  9  6 15  3 12 11
 6 10  5 12  5  1  1  3 16 14  3 15  6 13 15  5  2 17
false
```

19) Recursion start position d-1, d-1

```
 2 12  8 12 12  5  5  9 18 18  1 13  4  2  6  8 17  8  9
 4  5  8 10 14 16  7  4  9 17 15  0 13 12 12 10 13 14 13
 3  8  8  6  3 18 16  6  1 12  8 14 16 11 12  6 12  8 18
 5 18  6  6  3  5  8  9 16  9  7  5 16 10 18  3 10  5  5
 1  3 13  9 12 10  2 12 16 17 18 13  9 16 17 13 14 12  6
 5  2 18  4  4  9  2 16 13 10 15  1  4 11  9  7  7  5 18
11 15 12 11 15 18 13 11  1 13  6 16 10  8 12  2 14 10 18
10  4 11  9 15  5 13 10  4  8 11  4 15 11 10  7 11  7 15
11  7  1 17 17  8 18  1 11 10  1 10  6  7  6  5  3 13 12
11 14 17 17 10 16 17 12 11 16  3 18  6  6  5  9  4  7  2
 1  1 18  1  2 13  4  5  8 17 13 12  5  5 17  3 15 18  1
 1  8  2  8  1  2  6 10 16  8 11  5 14 12 16 18  7 18 18
 3 13 10 14 14 15  3 13 17 15  7  1  8  1 10 16 11 15 13
 3 14 14  9 11 12 16 16 12 14 14  3 11  6  1  1  7  1 16
12 16 11  2 15  2 15  9  9  4 17  9 18 10  8  2 17 17  6
15 16 12  4  6 15  3  1 17 10 17 12 15 10  9  4 13  3  7
17  4  4 10 11 14  3  3  4  4  5 10  7 18  3  7 14 11  5
13  4  3 18 13 14  1  4  6 16  1 17 16  7 16  1 10  5 15
 7  6 12  2 11  5  1 14 14 14  8 14  1 12  2  6  7  1 12
false
```

20) Recursion start position d-1, d-1

```
 5  9  3  8 13  2 12  0 11  3 10  3  4 18 16 14  2 16  5  6
 7  9 15  2  4 11 10 17 18  5 13 12  2  8 12  3  6  3  4 14
 6 10 13  6 16  5 18 18 14 16  3 15  1  5 12  1  1 11 13  9
 5 12 14  9  7 12  7 15 15  3  4  8 15  4 19 19  6 12 10 10
13 19  1 10 15  4 10  6 19 15  9  2 18  8 13  1  7  2  5  4
 5 13  8 16  2  2  4  3  2  7  3  2 13 15  8  6  7  1  8  5
 4 11  1 10 17  4  8  2  6 15  1  6 17 18  7  8  7 15 14 12
 9 13 18 15  5 18 12  5 15 14 17  1 14  6 19 15 14  8  7  9
 3  5  3  3 12 19 17  4 18 11 18  6  2  7 16  7  6  1 15  4
 6  8 11 15  1  6  2  3  2 18  7 19  2  9 19  1 12 17 13  9
16 13  2  7  5 12 17 10  7  1  9 15  9  9  7  9 13 13  6  5
 1 19 19  4 16 12  1 10 17 11 19 10 16  4  9 15 19 17 10 13
17  7  2 18  1  7 17 13  2 16 10  2 15 11 11 17  5  4 15  8
18  9  3 11  5 12 16 15  8 12  8  1 11  6 17  2  9 13  5 12
19 10 15  5 12  8 16 18  5  7  3 15  4  3  8  5  5  3  5 17
17 11 15  9  1 18  6  3 19  4 14 11  3  2 17 19  1  5 17  1
 9  2  4 19 12  7  6  3 10  1 18  6 13 13  9  8  2  2  8  6
16  4 11 12 17  9 11  1  2 12 14  4 14 13 19  8  4 15 12  1
 8  1  1  3 15  8  9 15 10 18 14  4 11  3  8 15 17 14 15 16
15  6  3  6  5  1  2 14  2  6  7 12  9 18 18 18  6 14 18  8
There is a solution!
true
```

1) Iteration start position 0, 0

```
 4  4  1  4  1
 1  1  1  3  3
 0  4  3  4  1
 4  1  4  2  4
 1  1  4  4  3
There is a solution!
true
```

2) Iteration start position 0, 0

```
 0  3  1  1  3
 4  4  3  4  4
 4  1  3  4  1
 2  1  4  4  2
 4  4  3  4  2
This start position is invalid!
false
```

3) Iteration start position 0, 0

```
 4  3  0  4  2
 4  3  4  2  4
 3  1  3  2  4
 4  4  2  4  2
 1  1  3  1  3
There is a solution!
true
```

4) Iteration start position 0, 0

```
 0  2  1  1  3
 4  2  4  3  1
 4  2  2  4  3
 3  4  3  4  4
 1  1  1  2  3
This start position is invalid!
false
```

5) Iteration start position 0, 0

```
 4  0  4  3  4
 1  1  1  1  2
 3  2  1  2  4
 3  1  4  3  3
 2  2  1  1  3
There is a solution!
true
```

6) Iteration start position 0, d-1

```
 3  4  0  1  2  5
 2  4  5  1  5  3
 1  1  5  2  2  1
 4  4  2  4  4  2
 4  3  3  1  3  3
 4  2  1  2  5  2
There is a solution!
true
```

7) Iteration start position 0, d-1

```
 6  0  3  3  2  3  4
 4  4  5  1  3  4  3
 3  4  4  6  4  5  1
 6  5  5  2  5  2  6
 1  3  4  3  2  4  3
 2  1  5  3  5  2  4
 1  5  1  5  3  6  5
No solution
false
```

8) Iteration start position 0, d-1

```
 6  0  7  3  2  6  2  4
 4  7  4  7  3  5  7  7
 6  3  7  7  2  7  5  2
 2  2  2  5  3  6  1  4
 3  3  1  1  3  5  6  1
 3  4  3  7  3  2  7  5
 1  1  1  2  6  6  1  1
 7  2  2  6  3  5  5  1
No solution
false
```

9) Iteration start position 0, d-1

```
 0  5  2  8  2  4  2  6  2
 7  8  2  6  6  1  2  8  7
 1  5  5  1  5  6  6  3  1
 8  1  7  4  4  7  7  8  8
 6  3  1  2  5  3  7  2  8
 5  2  5  2  4  5  7  6  5
 6  6  4  4  8  3  6  7  5
 6  4  8  6  1  5  6  8  4
 7  1  5  1  8  4  6  7  1
There is a solution!
true
```

10) Iteration start position 0, d-1

```
 2  9  1  7  2  7  1  7  1  5
 3  8  3  7  0  7  7  6  7  8
 6  9  7  5  7  8  7  7  7  5
 3  4  1  7  6  8  8  1  8  8
 8  9  4  8  6  2  8  1  9  4
 4  9  1  6  4  9  1  5  2  1
 8  1  4  7  3  9  8  7  5  6
 5  4  3  9  7  1  1  7  3  9
 9  4  5  1  5  9  5  3  1  6
 9  7  4  5  7  4  5  6  3  5
No solution
false
```

11) Iteration start position d-1, 0

```
 1  1  0 10  1  7 10  6  7  6  3
 9  5 10  5  5  4  5  9 10  9  6
 6  6  7  7  4  8  6 10  9  6  7
 3  7  6  1  3  9  6  1  9  9  5
 4  1  2  2  3  7  5  3  5  1  8
 2  5 10  3  3  1  9  4  1  2  1
 7  2  7  3  7  8  3  2  9  9  7
 3 10  2  2  1  4  7  3  4 10  8
 1  1 10  8  8  1  5  4 10  7  9
 9  9  2  9  5  6  2  8  7  1  9
10  1  8  2  6  6  9 10 10  4  6
There is a solution!
true
```

12) Iteration start position d-1, 0

```
10  1  6  6  5  2  5 11 10  4  1  0
 10  3  9  6  6 11  9  1 10  7  5 10
  5  3 11  9  7 10  8  1 10  2  6 11
  3 11  7  7 10  1  6  7 10  6  2  5
  9  3 11  9  9  1  7  9  2  7  5  5
  9  5  6  5  2  4  5 11  4 10  7  9
  2  3  7 10  1  5  3  7  6  8  3 10
  8  2  2  4  6  3  8  1  5  2 11  8
  5  7  8  9  1 11  3  3  6  6  6  9
  9 11  5  9  3 11  2  8  5  4  6  6
  9  6  3  7  4  8  2  4 10  2  1  1
  3  3  3  3  3  5 11  1  9  7  2  9
There is a solution!
true
```

13) Iteration start position d-1, 0

```
 8  7  7  6 10  4  3  6  6  2 12 11 12
 1 12  9  3 10  0  7  1 12  4 11  3 10
 5  5  8  6  9  3 11  2 12  3  5  6  4
10 10  4  1  1  2  4 11 11  7 10  7  4
 9 12  5  9  1 11  6 11  2 11 12  6  9
 4  7  8  2  5  3  4  6 12  2  2  2  9
 2  2 12  6  7  5  4  9  9  9 12 12  5
12  8  6  7  4  4  6  2  2  4  9  8  6
12  7  4  3  1  6  2  9  4  2 10  1  5
 5  4 11  9  9  5  4 10  3 10  4  3  7
 4 10  3  7  6  3  3 11  4 12  2 12 12
11  5  8  1 12  7  8  9 11 12  2  3  9
10  2  2  8  8  6  3 10  2  4 12  5  1
There is a solution!
true
```

14) Iteration start position d-1, 0

```
10  9 13 12 10  6  0  9 12 12 11 11  6  2
 5  6  4  5  6  1 12 13  5  2 13  1  5  9
 9  9 11  4  8  8  4  8  9 12  3 12  5 11
12 11  1  3  7 11  6  1  3  4  9  9 10 10
 9  3  1 12  7 10  3 10 11  4  3  8 12 13
 7 10  8  8  2 10  3  8  6 13 11  5 13  4
 7  4  6  2  7  4 10  2  2 10  5 10  5  4
 8  8 12  7 13  9  8  1  3 10 10 10  9  9
 4  6  2  8  7 11  3  1  1  5  4  9  8  1
 3  9  5  5  7  3  9  2  5  7 13 13 10 11
13  6 13 11  5  5  3  4  9  1  2  6  7  1
 1  9  7 13  2 11  8 11  9 13  5  5  9  4
 2  5 10  1 12  8  9  1  7  9  8 10 13  3
 3  5  1  5 13  7  6  6  8 11  9  1 12 12
There is a solution!
true
```

15) Iteration start position d-1, 0

```
 12  6  2  0 11  5  2 13 11  9  2  5  2  1  4
 10  9  3  1 13  4 14 11  4 14  3  4  6  6 14
 12 10 10  7 10 12  2  8 11 10 11  3 11  2 13
  7  6 14  3  8  6  9  2 10  5  3  6  1 12  6
  3  1  2  9 12 11  4 14 14  9  6 12 10  2  1
  2  8 11 10  5 13 12  2 13  1 10  6  8 12 13
 13  4  1 13  6 10 11 14  2  9  3  9 13  5  9
  5 12  1  6  2 14  4  6  7  3 13  2 13  4 13
  6  4  5  3  6  7  4 13  3 14  6  5  5  7  9
 11  5  9 10 12 11  5 11 12  3 11  6  8  6  6
 14  5  7 11  9  6  5 10 14  6 11 10  5  2 14
  6  1  5  5 12  5  7 14  4 12  8 12 13 12 11
 12  6 12  2 12 14 13 11  4 14 11 12  4  3 11
 12  7  2 13  9  9 14 10 10  4  9 11  6  7  8
 13 11  3 14 10  4  6 11  1  9  3  8 14 12 12
There is a solution!
true
```

16) Iteration start position d-1, d-1

```
  3 12 10  7  0 15  9  6  4 15 15  6 15 13 15 10
 10  5 15  9  2 14  3  8 11  2 15 11 11 14  9 14
  7 14 15 14  5  6  4  5  1  3  3 10  6 10 12 15
  3  7  3 14  7  1  8  4  6  7  8  3  7  3  8 10
 14 11  5  3  6  4  2  1 12  5  5  9  5 11 10  9
  4 12  8  1  5  6  4  5  7  3 10 12  5  8  9  1
  9  8  2  3  9  3  8 15 12  3  3 11  3  6 13 14
 12 11 11  9 15  9  2  7  4  9 15  8 13  3  7  6
  9 11  8  7 15  3  6 12  1  8  7  2  4  6 13  2
  1 12 12  3  5  3 15  6  3  9  5  2  2  4 12 13
  8 11 11  6 10 10  2  1  3 13 12 11 10  1  7  7
  5 11  9 13 14 11  3 14 11  6 13 11 13  9  8  6
  8  1 13  4  3 11  7 14 10  8 14 15  8  4  5  7
  5  7  9  1 11  7  7  6  8  2  8  6 13  3 12 12
 10  2  1  7  6  6  3  9  4  7  5 10 15  2 15 13
 10  9  8 13 10  3 11  3  6 15  5  1  3  7 13  9
There is a solution!
true
```

17)

```
  6 11  0 14 10 12 11 10 11  8  8 16 14 11  7 10  1
 12  4 10 14  3  8 11 15  1  5 10 14  3  9  1  6 14
 11  8  3  3  2  4  3 12  4 15  5  4  9  7 10  8 13
 13  4  1 14 11  3  2 13  1  4  3  6  2  9 16  4  9
 12 10 14 10  5 13  5  6  2  4  4 16 12  4  1 13  7
  3  9 10 10 14 16 15 13 13 12  8  4 11 12 14  9  6
  9 12 12  8 14 10  5 10 11  1 10 13 14  3 10 10  1
 16 11  9 13  7 16 11 16 10 14 15  8 13  7  4 11  9
  3  4  6  9  1  5 15 14 16 12 14  6  5  2  8  2 13
  3  1  6  2 16  3 14  8 12  9  4  1  2 16 13 12  8
  1  4  6 13  9  9 16  6  3  4  1 11  2  7  6  7 12
 13  9  9 11  9 15  7 15 15  6  5  7  3  7  7 15  8
 14 11  5 13 16 13  7 11  1  8 12  6 16  9 13 15 11
  8  1  5 11  8  6 11 14  4  8  7 13 16  2  9 13  9
 14  1  2  7 11  4 13  9 11 14  2  7 16 13  8 13 10
 13 16  3 16 11 10 11  3 11 13 11 12 10  6  9  8  2
 12 11  9 12  8  6  9  5  8 15  9  5 12  2  2 10  1
There is a solution!
true
```

18)

```
 14  4  8  8 11  0  2  2 13 13 16  6 11  2  4 10 13  3
 11  6  7  3  5  3  7  5 14  4 10 17 14 14 14 12  5  7
  2 10  6  3 10  3  1  4 17  3 13  8  4  1  4  9  7  5
 10 10  3  8  8  4  3  2  1 16 13 15  6 11 11 17 11  7
 13  6  4 10  1 15 12 11  1 11  9  4  3 12  8 14  5 17
  7 16  7 11  3  7 11  1 17  6  7  9 15  5  5 11 17 15
  8  4 13  4 10 12 13  7  5  3  8 10  6  7  3  9  1 15
 14 13  5 14 12  2 15 11  1 16  7 17 12 16 12 17 16 17
 14  7 14 15  7  3  1  7  7  2 16  7 15 12  5 15 16 15
  8 16 16 12  2 15  2  9  3  9  4  8 16 17 10  3  8  7
 13 17  5  5  6 12  8 12  3 14 16  4 16  8 16  8 16  6
  6 17  9 17 12  3 15  7 16  6  7 12 15  4  4  9 11 13
  5  5  8  4  6  6 11 13 12  7 11  7 10  7 11  4  6 13
 17  7 16 16  1 17 16  4 12 15 10 16  8  1 16  5 12 13
 17 11 17 14  6  5 16  5  7  6  1  7  6  3 16  7  1 16
 14  2  4  4  5  6  3  1  3  3 11 14 14  8 14  4 11  3
  6  1  9 12  8  7  1 17  4 12  2 14 15 13 16  5 14 17
 13 17 17 11  6  5  5 15  7  7 16  9  6  6  9  6 13  1
There is a solution!
true
```

19)

```
 13 10  2  3 11 11  3 13  5  6 18 12  8 12 15 12  9 15  3
  4  0 17 16  4  7 14  1 13 18 15  6  3  9  6  5  6 10  4
  9 14  2 16 10 16 15  5  8 13  7  9  4  1 14 16 12 18  9
  3  9 13 15 10  6  2  8 10 15 10  1  2 14 12 16  5 10  7
 15 18  4  5 16  7 14 15 16  9 15 12 11 13 17 13 14  6 18
  9  7  5 10 14 11  7  5  3 16  7  4 10  1  9 16  2  6 17
  7  2  2 17 12  6 17 13 15 14  7 15  2  8  5 12  9 13  5
  3  7  5 15  6 18  2  6  3 18  5 10  1 16 15 17 14  7 14
 13  4  2  9 16 17  8  2 15  3  8 10 14  7 13  3  6 15  1
 12 15  8  5  3  4 14 18 13 14 16  6 17 18 10 16 11 10 17
 16  3 13  7  4 14  4  4 10  2  9 10  1 16  9  9  5  6 16
  1  2  4  1 13 16 10 12 10 11  3  9  2  6 10 11 13 13  6
  5 15 18 12  5 15  6 18 18  9  6 16 12 11  7 18  4 18 12
  7 15 10 11  7  2  3  2 12  3 11 13  6 18  8 13  7  6 11
  6  3 11 14  7 10 18  7 18 15  1  4 18 10 15 17  1 13  2
 11 10 17 13 16 18  5  5  8  7  9 16  8 14 12 18  4 16  1
  7 17  9  6  1  4 17  6 11 17 17  6 12  3 13 14  7 15 16
 11 18  1  4 14 17  6  9  6  1 16 13 11  6  6  7  5  8 11
  2 17  6  1 12 17  8  7  1 17  5  8 10  7 11 18  1 10  7
No solution
false
```

20)

```
 17  8 12  7  3 15  8  6  9 14 10 10  9  7  2  9  9  5 17  5
 13  9  0 19 15 17 16 17  5  1  6  3  3 15 12  9 12 12  6 11
  7  6  5  3  9 12 19 12  3  6 19 11  9  9 10  6 16 17  3  9
 12 10  1 12  7  5  6 13  9  3 13 15  9 19  4  6  9  1 11  3
 12 15  3 14  6 12  5  5 15 12  1 16 19 10 11 12 16  3  7  4
 12  4  6  7 15  3 15  9 13  3 13 18 19 13 15 11  2  3 19  6
  8  4 10  7 14  3  6 12 19  5 17  1 18  6  5 12  6 18 13 15
 18 17 14 19 19 11 15 18  7 10 11 17  3 19  5  4 19  5 16 16
  9 19  2 17  8 14 14  5  7  3  7 12  1  1  9 12  4 15  8  8
  5  7 15 11 19 16 15  2 10 18  5 15  4 13  7  2  4  2  3  8
  5  4  6  6  8  9  7  2 16  4 18  9  2 11  3 11  5 18 19  5
 14 10  2  8  8  1  4 11  4 19  7  7 15  7 10  8  3 16 12  6
  2  1  8 16  4 10 13  7  7 18  7 16  5 14 16  2  3 15 18  8
  9  1  6 10  1 18 18  6  7 18  3  2 19 11 14  6  7 15  9  9
 11 14  6 12  5  5 11  8 16 15 18  6 16 15  4  8  1 13  9  6
  4 12 14 15 13 13 10  2  1 11  5  3  4 16 12 13  6 19 17 16
 18  9 13 18 16 18  5 19  1 12  6  4 12 13 11 15  6  7 19 18
 10 10 15  6 11  4 11  7  4 12  2  4  1 19 13 18  7 16 19 18
 14 14  4 15  7  5  8 14 13  4  8  9 12  3  8  3 17  2  2 15
  5 18 18 18 19 14 16 13  1 10  8  1  8 15  4  3  9 19 10  2
There is a solution!
true
```

e) Explain how one can detect unsolvable board configurations and whether there exists a way to speed up the execution time.

This can be done by finding all possible squares that can be visited on the board, in other words, all possible board configurations based on the game rules. If a square has already been visited, it is disregarded. To speed up execution time, one could check if a square has been visited before doing recursion or looping.

Pseudocodes

1) Recursive

```
Algorithm isValid(board, d, previous, row, column, check)
Input: board of size d x d, previous squares visited, starting row and column, and a check position
Output: validity of the board for the game

If check is 0 and board[row][column] is 0 then
        Print “This start position is invalid!”
        Return false
If check > 0 and board[row][column] is 0 then
        Print “There is a solution!”
        Return true

If previous[row][column] was visited
        Return false

previous[row][column] ← true
done ← false
temp ← 0

if column + board[row][column] < d then
        temp ← column + board[row][column]
        done ← isValid(board, d, previous, row, temp, check + 1)
        if done is true then
                return true
if column - board[row][column] >= 0 then
        temp ← column - board[row][column]
        done ← isValid(board, d, previous, row, temp, check + 1)
        if done is true then
                return true
if row + board[row][column] < d then
        temp ← row + board[row][column]
```

```
        done ← isValid(board, d, previous, temp, column, check + 1)
        if done is true then
                return true
if row - board[row][column] >= 0 then
        temp ← row - board[row][column]
        done ← isValid(board, d, previous, temp, column, check + 1)
        if done is true then
                return true

return false
```

2) Iterative

```
Algorithm isValid(board, d, previous, row, column)
Input: board of size d x d, previous squares visited, starting row and column
Output: validity of this board for the game

If board[row][column] is 0 then
        Print "This start position is invalid!"
        Return false

done ← false
rows ← new Stack
cols ← new Stack
push row in rows
push column in cols
temp ← 0
currentRow ← row
currentCol ← column

while done is not true
        if rows is empty then
                print "No solution"
                break from loop
        currentRow ← pop rows
        currentCol ← pop cols
        if previous[currentRow][currentCol] is true then
                continue to next loop
        previous[currentRow][currentCol] is true
        if board[currentRow][currentCol] is 0 then
                done ← true
                break from loop
        else do
                if currentCol + board[currentRow][currentCol] < d then
                        temp ← currentCol + board[currentRow][currentCol]
                        push currentRow in rows
```

```
                    push temp in cols
                if currentCol - board[currentRow][currentCol] >= 0 then
                    temp ← currentCol - board[currentRow][currentCol]
                    push currentRow in rows
                    push temp in cols
                if currentRow + board[currentRow][currentCol] < d then
                    temp ← currentRow + board[currentRow][currentCol]
                    push temp in rows
                    push currentCol in cols
                if currentRow - board[currentRow][currentCol] >= 0 then
                    temp ← currentRow - board[currentRow][currentCol]
                    push temp in rows
                    push currentCol in cols
if done is true then
        print “There is a solution!”
        return true
return false
```