Homework Sheet 11

Author	Matriculation Number	Tutor
Abdullah Oğuz Topçuoğlu	7063561	Maryna Dernovaia
Ahmed Waleed Ahmed Badawy Shora	7069708	Jan-Hendrik Gindorf
Yousef Mostafa Farouk Farag	7073030	Thorben Johr

Exercise 3

We will traverse the board using BFS and thats it. The board is a directed graph. The ladders and snakes create edges between the nodes and we have edges between consecutive nodes if there is no snake head at that node. Assuming the board data is given as two lists (snakes and latters) and a number n for grid size, we will first create a graph and then run BFS on it.
Pseudocode:

```
int SmallestNumberOfSteps(int n, list of (int, int) snakes, list of (int, int) ladders)
  // Create graph
  graph G
  for i from 1 to n do
    G.addNode(i)

  for i from 1 to n-1 do
      G.addEdge(i, i+1) // if there is a snake head, we will remove the edge below
  for each (i, j) in snakes do
    if G.hasEdge(i, i+1) then
      G.removeEdge(i, i+1)
    G.addEdge(i, j)
  for each (i, j) in ladders do
    G.addEdge(i, j)

  // BFS
  queue Q
  array visited // initially all false
  array distance // initially all infinity
  Q.enqueue(1)
  visited.add(1)
  distance[1] = 0

  while not Q.isEmpty() do
    current = Q.dequeue()
    for each neighbor in G.getNeighbors(current) do
      if neighbor not in visited then
        visited.add(neighbor)
```

```
            distance[neighbor] = distance[current] + 1
            Q.enqueue(neighbor)

    return distance[n]
```

Correctness:
In the graph representation the "the smallest number of steps to move from 1 to n" is equivalent to finding the shortest path from node 1 to node n. BFS is guaranteed to find the shortest path in an unweighted graph. Since each move from one node to another is considered as one step, BFS will correctly compute the smallest number of steps required to reach node n from node 1.

Running Time Analysis:
Creating the graph takes O(n + $|S|$ + $|L|$) time. Running BFS takes O(n + m) time, where m is the number of edges in the graph. In our case, m is at most O(n + $|S|$ + $|L|$) since each node can have edges to its next node, and additional edges from snakes and ladders. Thus, the overall time complexity of the algorithm is O(n + $|S|$ + $|L|$).