Explanation

Task 1 (a):

In this task, we take reading from the input file from where we get the vertices and the edges. Then we create a matrix and put specific value in the desired position.

Task 1 (b):

For this task, reading of the vertices and edges are taken at first. Later an empty dictionary with 0 to vertices number were kept as key. Later another reading was taken where 1st digit was considered as key. Then the 2nd values were put under the same key in the dictionary.

Task 2:

In this task we used a graph function to put the values in a dictionary accordingly to their keys. Later this dictionary was sent to BFS function. Where a node and their sub node is constantly visited while keeping a track of the visited and non visited node. Nodes were kept in a queue and first element was popped to visit it's neighbouring node.

Task3:

In this code, we use the graph function to build a dictionary. Then, dictionary is sent to Dfs function where a node and it's neighbour node is visited to the depth. After there is no neighbour node to visit, it returns to the original node to visit further neighbouring nodes.

Task4:

To check if there is any cycle in the graph, we used DFS function. Every time we visit a node we put True on the array to keep a track of it. If a node is visited and later comes back to it when True is given it sends the value of being cyclic.

Task5:

In this code, we check every possible path from start to end. We use BFS method to desired destination. Every time a node is visited, track has been kept in paths and visited array. If the desired location is not there we keep iterating. Path holds the possible locations and visited holds the previous locations. Through this we backtrack to find the shortest time. Shortest time will length of total nodes - 1.

Task 6 :

For this problem we used DFS. We checked if the row, column is in the bound or there is an obstacle or it is already visited, then we mark the path as visited and if there is a diamond we ~~add~~ increase our counting by 1. We use the DFS function to move up, down, left, right to find diamonds.