Task 1 (a)

Firstly, the code opens the file from input1a.txt and assigns the value. Then it opens the output file and assigns the value to f2. Then it reads all lines from T. Then it splits the first line of the input file using whitespace. Then it converts the a and b value into integer. adj_matr initializes a 2D List with a+1 rows and a+1 columns. Therefore, in the end it closes the files.

Task-1 (b)

Firstly, the code reads edge information from input1b.txt. Then it constructs an adjacent list and then it writes the adjacent list to the output.txt. Each line in the output file represents a node and adjacent nodes with their respective weight. In the end, it closes the file using output1b.txt.

Task - 2

The code firstly takes input from the file. The code basically performs a BFS traversal on an undirected graph represented as an adjacency list ~~start~~ starting from node 1 and writes the BFS path to output2.txt file. The BFS path represents the order in which nodes are visited during traversal.

Task - 3

The code reads data from input3.txt. It basically performs a DFS traversal on an undirected graph representing as an adjacency list starting from node 1 and writes the DFS path to output3.txt. The DFS path represents the order in which nodes are visited during traversal exploring as far as possible along each branch before backtracking.

Task-4

The code reads a file named input4.txt and performs a depth-first search (DFS) on an adjacency list to check if there is a cycle in the graph. The graph is represented as an adjacency list where the vertices are numbered from 1 to 'a' and 'b' edges are read from the file and added to the adjacency list. The DFS starts from the first node and explores it's neighbours. After the DFS is complete, the code checks the 'cycle' variable. Basically, the code determines whether a given graph contains a cycle. and writes the result to an output file.

Task-5

The code reads input from a file named input5.txt and performs a breadth-first search on an directed graph represented as an adjacency list. The code starts the BFS from the first node using a queue the containing tuples of the form. the code then writes the output

to an output file named output5.txt. the code fi
the shortest path from node 1 to a given destin
node d in an undirected graph using BFS.

Task - 6

Firstly, the code reads input from a input f
The code performs a DFS on a 2D grid to
the maximum diamond-shaped subgrid in the gr
The DFS is a recursive DFS function that
explores the neighbouring cells of a given cell
The main part of the code iterate through
the cells in the grid. It initialises a visit
to keep track of visited cells and calls t
function, It then update max-diam variable.
Finally, the code writes max-diam value
an output file.

Bonus - 1

The code reads input from a file. The code then finds the two nodes that are farthest apart in the graph., the nodes that have maximum distance between them. The farthest variable is update to store the nodes that have the maximum distance. After iterating, it writes the farthest node.

Bonus - 2

The code reads input from a file. Each case represents a connected graphs where the nodes are either Vampire or Lykan. The graph is represented using adjacency list. The main function takes a case number, the number of nodes and list of flights. and uses DFS. For each case, the code finds the maximum of vamp_e and lykan_e and writes it along with the case number to the output file.