WEEK -04

TASK 1:

PART- A:

*Read the data and formatted it into several list into a single list
*graph_maker() it takes two arguments , the arrays and the vertex
*The first loop creates a matrix and the second loop runs through the array and adds it into the matrix .
*graph_presenter() this makes the matrix arrays presentable by looping through it and it writes each line in the output note

PART-B:

*same data format as PartA
*at first adding empty lists in range of the size in a list'*the
*then by double loop adding the end and weight into the lists by making them a tuple
*Now for printing purpose adding them into a variable in a dictionary format
*returning the variable.

TASK 2:

*formatting all the data in a dictionary
*bfs(), takes two arguments graph and the source
*two empty list for queue and visited list
*appending the source to the queue
*while loop till queue is not empty
*at first it pops the queue
*then if it's not in visited it appends in visited and prints out
*then it checks all the connected nodes and adds it to the queue if it's already not visited

TASK 3:

*formatting all data in a dictionary
*dfs() takes 3 arguments graph, source and the visited list
*checks if the visited has been declared or not. Then creates an empty list.
*This implies a stack policy to track the nodes.
*then it appends the source into the visited list also prints out the source
*after that it recursively visits all the nodes connected below if it was not in the visited list.
*the given parameter in the source would be the connected nodes.

TASK 4:

*same formatting as Task2 and 3
*dfs_cycle_check() takes two arguments graph and nodes

*It uses an extra dictionary and a list to track the connection
*in the dictionary it caps all the nodes in boolean data to track the visited nodes
*and then the other list works as a stack
*we make each of the node true and appends to stack
*then by checking if the connected nodes are visited or not.
*If yes returns true or it checks if it's in the stack or not, which also returns the truth.
*And if both of the case does not match it removes the item from the stack and make the visited False and also returns False
*In the case of Truth it's "YES" or it's "NO".

TASK 5:

*it uses same logic as bfs but uses the visited list as a fact checker by using boolean data
*by checking each level when it reaches the destination it returns the path and jump
*else it filters the path by boolean logic list and appends it on the queue to check the further level

TASK 6:

*max_diamonds takes the graph and rows and columns as arguments
*then it uses nested loops to run through the matrix/graph
*then it checks if it hits obstacle or not
*the it creates a fact checker boolean visited list
*then it compares the max_diamonds every time for each path.
*Now for the path it uses dfs()
*dfs uses the same logic and dives into recursive ways by checking each point's left,right,up and down.
*then it returns it to the main function.