206766677\_omer.apter@msmsail.ariel.ac.il

**מטלה 1- מחלקות ושיטות**

**Graph:**

VARIABLES: (are static for encapsulation)

* **Nodes\_num:** stores the number of nodes in graph.
* **Num\_edges:** stores the number of edges in graph.
* **graph\_loaded:** is true when this graph has already done a successful loadGraph method.
* **actual\_graph:** stores the weights and connections as an array that is representing a double array (as seen in lecture 7 in system programming 1)**.**

METHODS:

* **Load Graph**: Load a graph represented by a square matrix.
* **Set Edge**: Set the weight of an edge between two vertices.
* **Get Edge**: Get the weight of an edge between two vertices.
* **Print Graph**: Print information about the graph, including the number of vertices and edges.
* **get\_num\_of\_nodes:** **Get the number of nodes in the graph.**

**Algorithms:**

METHODS:

* **run\_bellman\_ford:** Runs the Bellman-Ford algorithm to find the shortest path. return -1 if negative cycle else return 0 and fills the d and prev arrays with the correct information according to Bellman ford.
* **Relax:** Updates the d and prev arrays during Bellman-Ford relaxation.
* **dfsCircleFind:** Uses depth-first search (DFS) to find a cycle in the graph. Returns a string representing a cycle in the graph. If no cycle return 0.
* **dfsCircleVisit:** Visits nodes in a DFS to detect cycles. Returns a node where a circle starts if there is a circle else returns the number of nodes (serves the same purpose as -1 or Null).
* **isConnected:** Checks if the graph is connected.Uses BFS and union find where starting from node 0 "merges groups" with each of its child nodes and iterates on all nodes, returns true if connected else false.
* **shortestPath:** Finds the shortest path between two vertices. Uses Bellman-Ford Algorithm to find the shortest path, if exists return true and print the path from start to end, else return false.
* **isContainsCycle:** Detects if the graph contains a cycle.
* **isBipartite:** Determines if the graph is bipartite.using BFS and answer from tirgul 4 of algo 1 if Graph g is Bipartite return a string showing the 2 groups else return "0".
* **negativeCycle:** Detects if the graph contains a negative cycle. Uses Bellman-Ford Algorithm on graph returns 1 if graph contains a negative cycle else return 0.