**Graph and Algorithms project**

This project includes two main classes, Graph and Algorithms.

Each of the classes serving a different purpose in handling and performing various algorithms on graphs.

**Graph class:**

The class represent a graph and provides methods for loading and printing a graph.

**Methods**:

* **void loadGraph(vector<vector<int>> gr)()-** the function gets an adjacency matrix and loading it to the graph. It also checks if the graph is directed and if the graph is weighed.
* **void printGraph()-** the function prints the number of vertices and edges the graph has.

**Algorithm class**:

The class provides various algorithms to perform on graphs including checking for connectivity, finding the shortest path between two vertices and more.

**Methods:**

* **bool isConnected(Graph gr**)- the function checks if the graph is connected by using bfs algorithm.If the graph is connected, the function will return true. Otherwise, it will return false.
* **string shortestPath(Graph gr, size\_t start, size\_t end**)- the function splits the shortest path check for two functions. If the graph is weighed, the function will call the shortestPathWeighed function. otherwise, it will call the shortestPathUnweighed function. If the path exist, it will return a string with the shortest path, otherwise, it will return -1.
* **string shortestPathWeighed(Graph gr, size\_t start, size\_t end**)- the function checks for the shortest path between the two vertices by using Bellman Ford algorithm.
* **string shortestPathUnweighed(Graph gr, size\_t start, size\_t end)-** the function checks for the shortest path between the two vertices by using bfs algorithm.
* **string isContainsCycle(Graph gr)-** the function checks if the graph contains cycle by calling the checkIsContainsCycle() function.

If it does, the function will return a string of the vertices in the cycle, otherwise, it will return "0".

* **string checkIsContainsCycle(size\_t v, vector<bool> &visited, vector<size\_t> &visited, stack<int> &cycle,vector<vector<int>> &graph )-** the function get parameters from the IsContainesCycle() function and checks if it is by using dfs algorithm.
* **string negativeCycle(Graph gr)-** the runs Bellman Ford algorithm on the graph n times(n means number of edges). If at the n's time something changed, the function will return a string "The graph contains negative cycle". Otherwise, it will return "The graph doesn't contains negative cycle"
* **string isBipartite(Graph gr)-** the function checks if the graph is bipartite by using bfs algorithm. The function colors every vertex it gets, if a vertex has the same color as one of his adjacencies, the function will return "0". Otherwise, it will return the partition for two sides and which vertex each of the sides contains.