PRACTICAL WORK 3

- **UI**

```
void uiFindLowestPathBetweenTwoVertices(DirectedGraph& graph) {
                    string filename;

                    cout << "The filename is = ";
                    cin >> filename; //reading the filename
                    readFromFile(graph, filename); //reading the graph from file

                    int startVertex, endVertex;
                    cout << "The start vertex is = ";
                    cin >> startVertex; //reading the start vertex
                    cout << "The end vertex is = ";
                    cin >> endVertex; //reading the end vertex

                    vector<vector<int>> cost = vector<vector<int>>();
                    vector<vector<int>> next = vector<vector<int>>();
                    GenerateCostMatrixOfADirectedGraph(graph, cost, next); //generate the cost matrix of the graph
                    FloydWarshall(graph, cost, next); //computing the shortest pahts from all the nodes to all the nodes in graph
                    if (cost[startVertex][endVertex] != inf) { //if exists at least one path between the start vertex and end vertex
                              cout << "The lowest cost path has the cost " << cost[startVertex][endVertex] << "\n";
                              uiWritePath(next, startVertex, endVertex);
                    }
                    else
                              cout << "It doesn't exist a path between " << startVertex << " and " << endVertex << "\n";
          }
```

- **BackEnd**

```
void GenerateCostMatrixOfADirectedGraph(DirectedGraph& graph, vector<vector<int>>& cost, vector<vector<int>>&
next) {
          cost = vector<vector<int>>();
          next = vector<vector<int>>();

          int n = graph.getNumberOfVertices();
          for (int i = 0; i < n; i++) {
                    cost.push_back(vector<int>()); //add an empty element
                    next.push_back(vector<int>()); //add an empty element
                    for (int j = 0; j < n; j++) {
                              cost[i].push_back(inf); //we suppose there is no edge between i and j
                              next[i].push_back(-1);
                              if (graph.isEdge(i, j)) { //if we find an edge we change the cost and the successor
                                        cost[i][j] = graph.getCostOfAnEdge(i, j);
                                        next[i][j] = j;
                              }
                    }
          }
}
```

```
void FloydWarshall(DirectedGraph& graph, vector<vector<int>>& cost, vector<vector<int>>& next) {
	vector<int>path = vector<int>();

	int N = graph.getNumberOfVertices(); //get the number of vertices in the graph

	for (int k = 0; k < N; k++) { //we parse the graph N times, so that we can construct a path between any 2 vertices
		for (int i = 0; i < N; i++) {
			for (int j = 0; j < N; j++) {
				if (cost[i][j] > cost[i][k] + cost[k][j]) { //if we have found a lowest cost path between the
vertices i and j
					cost[i][j] = cost[i][k] + cost[k][j]; //we update the cost
					next[i][j] = next[i][k]; //we update the successor
				}
			}
		}
	}
}

//the function which writes the walk
void uiWritePath(vector<vector<int>>next, int startVertex, int endVertex, DirectedGraph& graph) {
	//cout << startVertex << " ";
	int cost;
	while (startVertex != endVertex) {
		cost = graph.getCostOfAnEdge(startVertex, next[startVertex][endVertex]);

		cout <<  "((" << startVertex << ", " << next[startVertex][endVertex] << "), " << cost << ") ";
		startVertex = next[startVertex][endVertex];

	}
	cout << endl;
}
```