Homework 1 – Documentation

**Specification**

We shall define a class named Graph representing a directed graph.

The class Graph will provide the following methods:

Graph (nr\_of\_vertices=0, nr\_of\_edges=0)

Constructs a graph with n vertices and with m arcs.

get\_vertices\_number()

Returns the number of vertices

check\_if\_vertex(vertex)

Checks if a given number represents a vertex in the graph.

add\_vertex(vertex)

Adds a vertex to the graph. Precondition: the vertex must not exist in the graph already.

remove\_vertex(vertex)

Removes the given vertex from the graph.

deg\_in(vertex)

Gets the in degree of an existing vertex.

deg\_out(vertex)

Gets the out degree of an existing vertex.

get\_edges\_number()

Gets the number of edges.

check\_if\_edge(vertex1, vertex2)

Checks if there is an edge from vertex 1 to vertex2.

add\_edge(vertex1, vertex2)

Inserts an arc from vertex1 towards vertex2. Precondition: there must be no arc from v1 to v2.

remove\_edge(vertex1, vertex2)

Removes the edge from vertex1 to vertex2.

get\_edge\_cost(vertex1, vertex2)

Gets the cost of the edge from vertex1 to vertex2.

set\_edge\_cost(vertex1, vertex2)

Sets the cost of the edge from vertex1 to vertex2.

copy()

Makes a copy of the current graph.

The rest of the member functions are:

read\_file(file\_path)

Reads a graph from a file.

write\_file(file\_path, graph)

Writes the current graph to a file.

random\_graph(vertices\_number, edges\_number)

Generates a random graph.

**Implementation:**

Class graph will have the following data members:

nr\_of\_edges

The number of edges

nr\_of\_vertices

The number of vertices

\_\_din

The in dictionary of the graph

\_\_dout

The out dictionary of the graph

\_\_vertices

The list of vertices

\_\_costs

The dictionary which matches edges to their cost.