ADTs Graph

G = (V, E), where V is a set of vertices, and E is a set of edges

$\{inv:\ There\ cannot\ be\ two\ vertexes\ with\ the\ same\ value\ on\ the\ Graph.\}$

Primitive Operations:

-	Graph	Graph	→	Graph
-	addVertex	Graph x Value	→	Graph
-	addEdge	Graph x Value x Value x String x Int	→	Graph
-	deleteVertex	Graph x Value	→	Graph
-	deleteEdge	Graph x Value x Value x String	→	Graph
-	search Vertex Index	Graph x Value	→	Int
-	searchEdge	Graph x Value x Value x String	→	boolean
-	dijkstra	Graph x Value x Value	→	Int []
-	DFS	Graph	→	Graph

Graph()

"Create a new Graph"

{ pre: TRUE }

{ post: a Graph is created}

addVertex(Graph, v)

"Adds vertex "v" to the graph G"

$\{pre:\ True\}$

{ post: The vertex is added to the Graph }

addEdge(a, b, w**)**

"Adds a edge from vertex "a" to vertex "b" of weight "w" **"**

$\{pre:\ pre:\ a\ and\ b\ must\ belong\ to\ the\ set\ of\ vertices\ of\ the\ graph\}$

$\{post:\ :\ An\ edge\ connecting\ a\ with\ a\ is\ created\ in\ the\ graph\ \}$

deleteVertex(v**)**

"Deletes the vertex "v" from the Graph "

$\{pre:\ u\ must\ belong\ to\ the\ set\ of\ vertices\ of\ the\ graph\ G\ \}$

{ post: : The vertex is removed from the graph G }

deleteEdge(u, v, id)
"Removes edge(u,v) with identification "id" from the graph "

$\{pre\colon\ There\ must\ be\ an\ edge\ between\ u\ and\ v\}$

{ post: The edge is removed from the graph G}

searchVertexIndex(a)
"Searches the index in the graph of vertex "v" "

$\{pre\colon\ vertex\ "u"\ must\ be\ part\ of\ the\ graph\}$

$post\colon\ i \in \aleph\ \wedge\ i \geq 0$

searchEdge(a, b, id)
"Checks if there is an edge between two nodes in the graph "

$\{pre\colon\ [(a\ \wedge\ b)\ must\ be\ part\ of\ the\ graph]\ \wedge\ [there\ must\ be\ an\ edge\ between\ (a\ \wedge\ b)]\}$

$post\colon\ (true\ if\ there\ is\ an\ edge\ between\ a\ \wedge\ b)\ \vee\ (false\ if\ there\ is\ not\ an\ edge\ between\ a\ \wedge\ b$

Dikstraj(a, b)
"Finds the shortest path from vertex a to b in the graph. "

$\{pre\colon\ [(a\ \wedge\ b)\ are\ part\ of\ the\ Graph]\ \wedge\ [there\ is\ an\ edge\ between\ (a\ \wedge\ b)\ ]\}$

{ post: String chain with values in ascending order}

DFS( )
"Explore in depth by visiting all the neighbors of a vertex in the graph G. "

$\{pre\colon\ True\}$

{ post: The distance and time of discovery of each vertex in the graph G is determined.}