```
class Graph(object):


def __init__(self, graphIn = {}, graphOut = {}, costs = {}):

       Constructor
       # all nodes are integers
       Input:
       graphOut - dictionary - keys are nodes, each element is
          a list of destinations
       graphIn  - dictionary - keys are nodes, each element is
             a list of sources
       costs - dictionary - keys - tuples (source,
             destionation) of nodes, element - cost (integer)



    def checkIsNode(self, node):

        Checks if a node is in the graph or not
        Input: node - integer
        Output: True if it in the graph false otherwise



    def getNumberOfNodes(self):

        Returns the number of nodes
        Input: none
        Output: integer



    def parseNodes(self):

        Returns a list of all nodes in the graph
        Input: none
        Output: a list of nodes



   def getInDegreeOfNode(self, node):

        Returns the number of edges that have the given node
          as destination or False if it is not in the graph
        Input: node - integer
        Output: positive integer or False
```

```
def parseOutBoundEdges(self, node):

     Parses the outbound edges of a given node
     Input: node - integer
     Output: list of tuples of integer (source,
          destionation, cost)


def getOutDegreeOfNode(self, node):

     Returns the number of edges that have the given node
          as source or False if it is not in the graph
     Input: node
     Output: positive integer or False


 def parseInBoundEdges(self, node):

     Parses the inbound edges of a given node
     Input: node - integer
     Output: list of tuples of integer (source,
          destionation, cost)



 def addNode(self, node):

     Adds an isolated node to the graph if t is not already
          in the graphs
     Input: node - integer


 def removeNode(self, node):

     Removes a node and all associated edges if it is in
          the graph
     Input : node - integer



 def checkIsEdge(self, source, destination):

     Checks if an edge given by its source and destination
          is in the graph
     Input: source, destination
     Output: true or false
 def addEdge(self, source, destination, cost):

     Adds a certain edge given by its source and
          destination and sets its cost if it is not
          already in the graph
     Input: source, destination, cost - integers
     Output: None



 def removeEdge(self, source, destination):

     Removes a certain edge given by its source and
          destination if it is in the graph
     Input: source, destination
     Output: None




def modifyEdge(self, source, destination, newCost):

     Modifies a certain edge given by its source and
          destination with then new cost if it exists
     Input: source, destination, new cost - integers


def copyGraph(self):

     Creates a static copy of he graph
     Output: graph


def readFromFile(self, fileName):

     Reads a graph from a file
     Input: the name of a file formatted as follows:
     first line numberOfnodes numberOfEdges
     next numberOfEdges lines triplets <node1 node2 cost>




def writeToFile(self, fileName):

     Writes a graph to a file
     Input: the name of a file
```