```
int nrNodes();
// returns the number of nodes
int nrArcs();
// returns the number of arcs

std::pair<std::map<int, std::vector<int>>::iterator , std::map<int, std::vector<int>>::iterator>
setOfNodes();
// returns a pair of 2 maps that contain the nodes and their arcs so the graph can be crated

bool isNode(int v);
// returns true if the value v is a node, false if not

bool isArc(int v1, int v2);
// returns true if the v1 and v2 make a valid arc, else it returns false

void addArc(int v1, int v2, int cost);
// adds an arc between the v1 and v2 nodes with the cost 'cost'

int inDegree(int x);
// returns the degree of inbound arcs from the node x

int outDegree(int x);
// returns the degree of outbound arcs from the node x

std::pair <std::vector<int>::iterator, std::vector<int>::iterator> inboundArc(int x);
// returns a pair of the vector containing all inbound arcs from the node x

std::pair <std::vector<int>::iterator, std::vector<int>::iterator> outboundArc(int x);
// returns a pair of the vector containing all outbound arcs from the node x
```

```
void changeCost(int x, int y, int val);
// changes the cost of the arc between the nodes x and y

int getCost(int x, int y);
// returns the cost of the arc between the nodes x and y

void removeArc(int x, int y);
// removes the arc between the nodes x and y

void addNode(int val);
// adds a node val

void removeNode(int val);
// removes the node val

std::string toString();
// returns the graph in a string
```