HOMEWORK 1

We will build a simple graph data structure. It will be composed of node objects. Read about directional and nondirectional graphs. We will build a non-directional graph in this work.

Q1) node1.cpp class which has the following properties:

- id:integer
- An array of node pointers where array is fixed in size, 5.

a) Implement setter and getter functions for "id". Do simple tests in main function.

b) Implement **bool overwriteNeighborNode(index i, Node n)** member function. If you successfully overwrite the node at the index i, return true, else return false. Do simple tests in main function.

c) Implement **Node getNeighborAtIndex(index i)** member function. You should return the node at index i. Do simple tests in main function.

d) In the main function:

i) Create a node where id=1

ii) Create a node where id=2

iii) Create a node where id=3

iv) Add node-1 and node-2 as neighbors to each other both at index=0

v) Add node-2 and node-3 as neighbors to each other both at index=0

vi) cout<<node1.getNeighborAtIndex(0).getNeighborAtIndex(0).getId(); ?

vii) cout<<node2.getNeighborAtIndex(0).getNeighborAtIndex(0).getId(); ?

viii) Given that graph is not a directional graph(i.e. Edges are not directional), Is the resulting graph consistent?

Q2) Copy node1.cpp to node2.cpp.

For node2.cpp:

- remove overwriteNeighborNode, getNeighborAtIndex functions.
- Set max edge count to 1000, i.e., array size.

a) Implement a constructor with which you can instantiate a node, assign an id directly.

b) Implement **addNeighbor(Node n)** member function. You will need to do some assumptions, put them in your source code as comments. Hint: You may traverse through the node* array, find the first empty(i.e. NULL) spot and add the neighbor's pointer there.

c) Would using a friend function help us in this case? Given that our graph is non-directional you need to add nodes to each other, a two step process. Can you write a single friend function which sets both nodes as neighbors(no implementation, just explain)

d) Implement **bool removeNeighbor(Node n)** function which returns false if the neighbor cannot be found. Describe your logic in the comments.

e) What are the limitations of having a constant array size? Advantages, Disadvantages? How can we improve the design?