1. Give definitions of the following:
   a. - Node/Vertex
   b. - Edge
   c. - Neighbors
   d. - Adjacency list.

2. Write the code for a 3-D point class that has an \_init\_ function and an \_str\_ function that returns the string in the form \(x,y,z\)

3. Given the following Code, draw the resulting graph from calling the function \"makeGraph()\".

   class Node:
     __slots__ = ('name', 'neighbors')
     def __init__(self, name):
       __init__ : Node * String -> None
       Constructs a node object with the given name and no neighbors.
       self.name = name
       self.neighbors = []
     def makeGraph():
       loadGraph : String -> dict(String:Node)
       myGraph = {}
       if 'A' not in myGraph:
         nodeA = Node('A')
         myGraph['A'] = nodeA
       if 'B' not in myGraph:
         nodeB = Node('B')
         myGraph['B'] = nodeB
       if 'C' not in myGraph:
         nodeC = Node('C')
         myGraph['C'] = nodeC
       if 'D' not in myGraph:
nodeD = Node('D')
myGraph['D'] = nodeD
if 'W' not in myGraph:
    nodeW = Node('W')
    myGraph['W'] = nodeW

nodeA.neighbors = [nodeB, nodeC, nodeD]
nodeB.neighbors = [nodeD, nodeW]
nodeC.neighbors = [nodeA]
nodeD.neighbors = [nodeA, nodeB]
nodeW.neighbors = [nodeB]

4. Show the adjacency list for the graph in problem 3

5. Perform the following Depth First Search (DFS) for the graph (traverse in ascending order):

   a. start = 1 goal = 5

   b. start = 2 goal = 7

   c. start = 1 goal = 10