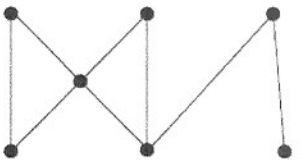
Tutorial 2 - Trees

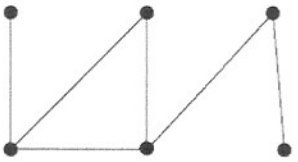
Question 1: Which of the following graphs are not trees? Give reason(s) for your answer.

a)

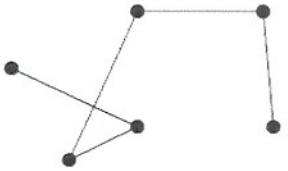
b)



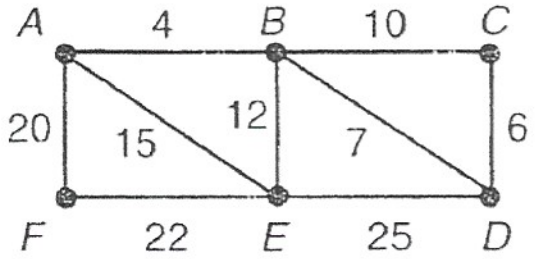
c)



d)



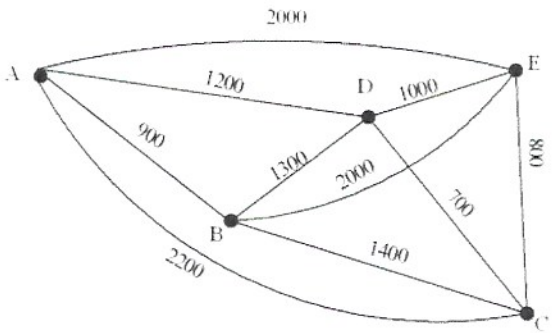
Question 2: Write down the weight matrix for the weighted graph given below:



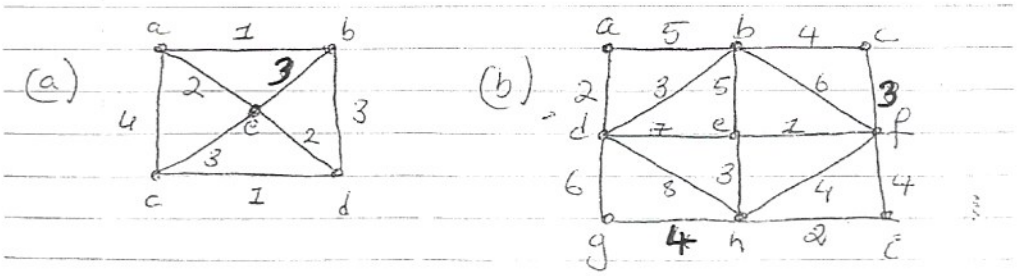
Question 3:

a) What is a Minimal Spanning Tree (MST) of a connected weighted graph?

b) Use ***Prim's* algorithm** to design a minimum cost communications network connecting all the computer buildings represented by the graph below. Show all your work.



Question 4: Use **Prim's algorithm** to find the Minimal Spanning Tree (MST) for the weighted graphs below:

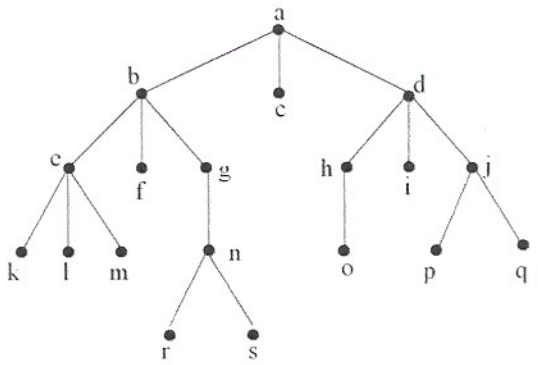


[redacted]

Question 6:
What is a **Rooted Tree** Structure? In your answer provide an example of a **Binary Tree** representation and explain the following concepts: *root node*; *parent node*; *child node*; *leaf node*.

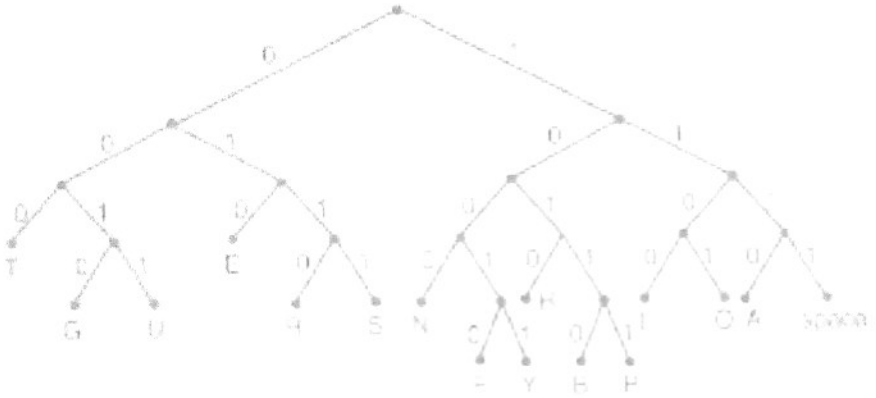
Question 7: Answer the following questions about the rooted tree below

- Which vertex is the **root**?
- Which vertices are **children** of *d*?
- Which vertex is **parent** of *g*?
- List all the vertices that are **leaf** nodes.
- Draw the **left subtree** of this tree that is rooted at *b*.
- Draw the **right subtree** of this tree that is rooted at *d*.



Question 8:

The binary rooted tree shown below can be used to encode and decode English text according to the *Huffman code*. The sequence of edges from the root to any letter yields the binary code for that letter. Note that the number of bits varies from one letter to another.



a) Decode 00100110 11101011 11010111 10001010 01011010 11010011 11111100 01111111 10010001 11000.

(The coded message is grouped into strings of eight bits to enhance readability only; the grouping has no other significance)

b) Encode 'TO BE OR NOT TO BE'

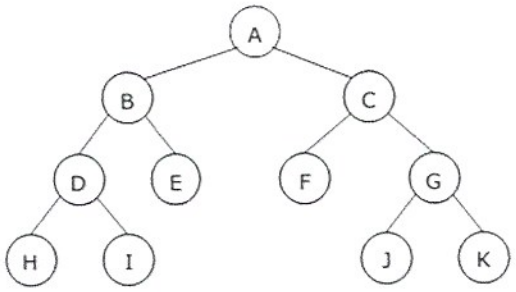
Question 9:

i) Build a **Binary Search Tree** for the following data: 50 30 20 40 70 60 80

ii) Perform **Breadth First Traversal** and **Depth First Traversal** on the constructed BST

Notes: A Binary Tree Data Structure is a hierarchical in which each node has at most two children, referred to as the left subtree and the right subtree. It is commonly used in computer science for efficient storage and retrieval of data, with various operations such as insertion, deletion and traversal.

For example:



To build a Binary Search Tree (BST):

1. Start with an empty tree
2. Insert each number from the list into the BST using the following logic:
    - If the tree is empty, the first number becomes the root
    - For each subsequent number:
        - If it's smaller than the current node, insert it into the left subtree
        - If its larger than the current node, insert it into the right subtree

Insert numbers, one by one, to construct the BST.

Breadth First Traversal explores nodes level by level, left to right at each level

Depth First Traversal explores as far as possible along each branch, use a stack data structure to backtrack.