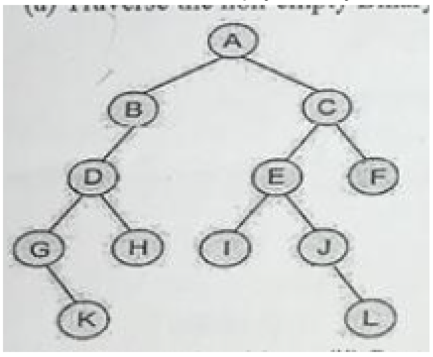
1. Briefly explain what data structure means and vie real life illustrations to buttress your points.
   A data structure is a way of organizing and storing data to perform operations efficiently. It defines the relationship between the data and the operations that can be performed on the data. In real life, think of a data structure as a way you organize and store information. For example:
   - **Phonebook:** A phonebook organizes names and phone numbers for quick retrieval.
   - **Library Catalog:** A library catalog organizes books by categories and author names for easy location.
   - **Filing Cabinet:** Files organized in folders and drawers to facilitate easy access.
2. Differentiate between linear and non-linear data structures, providing relevant examples of each data structures.
   - **Linear Data Structures: Organizes data in linear and sequential manner.**
     - **Arrays:** An ordered collection of elements with a fixed size. *Example: List of students in a class.*
     - **Linked Lists:** Elements are stored in nodes, each pointing to the next node. *Example: A chain of paperclips linked together.*
   - **Non-Linear Data Structures: Elements are connected in a complex branching non-linear manner**
     - **Trees:** A hierarchical structure with a root node and branching nodes. *Example: Organizational hierarchy in a company.*
     - **Graphs:** A collection of nodes connected by edges. *Example: Social networks with people as nodes and friendships as edges.*
3. Explain the concept of queue data structure using any 4 analogies of your own
   A queue is a data structure that follows the First-In-First-Out (FIFO) principle. Insertions and deletions occurs at different ends of the data structure. Imagine queues in real life:
   - **Grocery Store Checkout:** People stand in line, and the first one to arrive is the first to leave.
   - **Print Queue:** Documents sent to a printer are processed in the order they are received.
   - **Ticket Counter:** Customers waiting for tickets are served in the order they arrived.
   - **Elevator Queue:** People waiting for an elevator board in the order they arrived.
4. Data structures are normally classified into two types, namely primitive and non-primitive data structures. Explain with examples
   - **Primitive Data Structures:**
     - **Integer:** Represents whole numbers (e.g., 5, -10).
     - **Float:** Represents real numbers with decimals (e.g., 3.14).
     - **Character:** Represents a single character (e.g., 'A', '1').
   - **Non-Primitive Data Structures:**
     - **Arrays:** An ordered collection of elements.
     - **Structures:** Groups different data types under a single name.
     - **Pointers:** Stores the memory address of another variable.
5. Applying the FIFO principle, complete the table below:

Action	Contents of queue Q after Operations	Return Value
Initialize(Q)	-	-
Add(X,Q)	X	-
ADD(Y,Q)	X,Y	-
ADD(Z,Q)	X,Y,Z	-
Remove(Q)	Y,Z	X
Add(A,Q)	Y,Z,A	-
Remove(Q)	Z,A	Y
Remove(Q)	A	Z

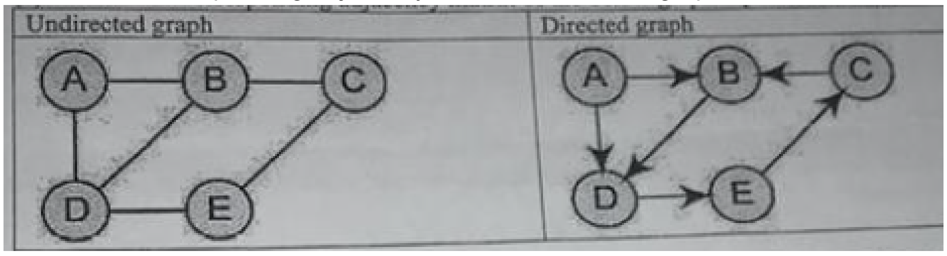
Action	Contents of queue Q after Operations	Return Value
Remove(Q)	-	A

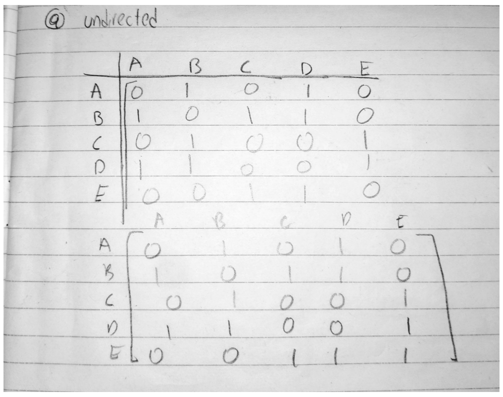
6. Traverse the non-empty Binary tree below and give the node order



1. Pre-order algorithm: A, B, D, G, K, H, C, E, I , J , L, F
2. Post-order algorithm: K, G, H, D, B, I, L, J, E, F, C, A
3. In-order algorithm: G, K, D, H, B, A, I, E, J, L, C, F

7. Discuss the following terminologies in Tree Data Structure
    1. Binary tree: A binary tree is a tree data structure in which each node has at most two children, referred to as the left child and the right child. These children are also roots of binary subtrees. Binary trees are commonly used in computer science and have applications in various algorithms and data storage structures.
    2. Root node: The root node is the topmost node in a tree hierarchy. It is the starting point for traversing the tree. In a binary tree, the root node is the only node that has no parent. It serves as the anchor point for the entire structure.
    3. Leaf node: A leaf node, also known as a terminal or external node, is a node in a tree that has no children. In a binary tree, a leaf node is a node without a left or right child. Leaf nodes are the endpoints of the tree branches.
    4. Degree of a node: The degree of a node in a tree is the number of children it has. For a binary tree, a node can have a degree of 0 (leaf node), 1 (one child), or 2 (left and right children). The degree of the tree is the maximum degree among all nodes in the tree.
    5. Similar binary tree: Two binary trees are considered similar if they have the same structure, meaning they have the same arrangement of nodes and the same connectivity. The values stored in the nodes might be different, but the overall tree structure is identical.
8. Perform the following operations
    1. 10's complement of 9856_{10} = 144
    2. 9's complement of 9856_{10} = 143
    3. 2's complement of 1101_2 =0011
    4. 1's complement of 1101_2 = 0010
9. Given an expression, $\exp = ((x + y) - (a * b))\%((e^f)/(u - v))$ construct the corresponding binary tree
10. Provide the corresponding adjacency matrix of the below graph

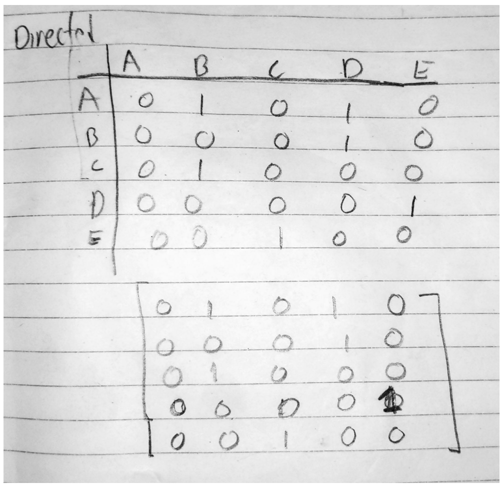




ⓐ undirected

$$\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & 0 & 1 & 0 & 1 & 0 \\ B & 1 & 0 & 1 & 1 & 0 \\ C & 0 & 1 & 0 & 0 & 1 \\ D & 1 & 1 & 0 & 0 & 1 \\ E & 0 & 0 & 1 & 1 & 0 \end{array}$$

$$\begin{array}{cc} & \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{array}{c} A \\ B \\ C \\ D \\ E \end{array} & \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix} \end{array}$$



Directed

$$\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & 0 & 1 & 0 & 1 & 0 \\ B & 0 & 0 & 0 & 1 & 0 \\ C & 0 & 1 & 0 & 0 & 0 \\ D & 0 & 0 & 0 & 0 & 1 \\ E & 0 & 0 & 1 & 0 & 0 \end{array}$$

$$\begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

11. Continuously perform the operation below using the List,

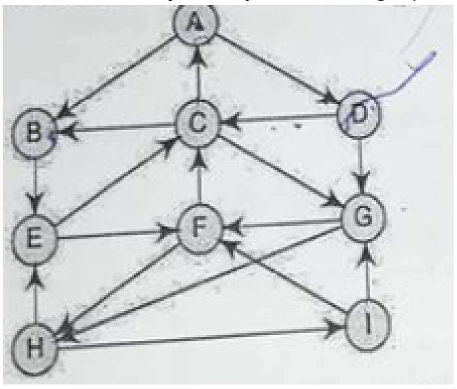
A	B	C

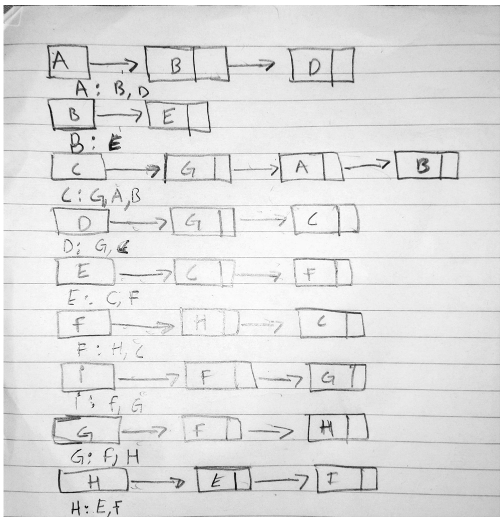
```
1. Add(1,X,L)
2. Set(2,Z,L)
3. Remove(Z,L)
4. Get(2,L)
5. IndexOf(X,L)
```

Operation	Data	Return
Add(1,X,L)	A, X, B, C	
Set(2,Z,L)	A, X, Z, C	
Remove(Z,L)	A,X,C	Z
Get(2,L)	A,X,C	C
IndexOf(X,L)	A,X,C	1

1. Enumerate any 5 operations performed on Stack data structure
   - Initialize: Initializes an empty stack
   - Pop: removes the last element in the stack
   - Push: Add an element to the top of the stack
   - IsEmpty: Checks if the stack has no elements inside it
   - IsFull: Checks if the stack if full.

2. Provide the adjacency list of the graph below





A → B → D
A : B, D
B → E
B : E
C → G → A → B
C : G, A, B
D → G → C
D: G, C
E → C → F
E : C, F
F → H → C
F : H, C
I → F → G
I : F, G
G → F → H
G: F, H
H → E → I
H: E, F

3. Create a binary search tree using the following data elements 45, 39, 56, 12, 34, 78, 32, 10, 89, 54, 67, 81.
4. Convert the following infix expressions into postfix expressions
    1. $(X + Y) * (A + B)$ = $XY + AB + *$
    2. $(A + B)/(C + D) * (D - E)$ = $AB + CD + /DE - *$
    3. $(I + J)^Z$ = I J + Z ^
5. Why is adjacency matrix called bit matrix

   The term "adjacency matrix" refers to a mathematical representation of a graph using a square matrix. In the context of graphs, an adjacency matrix is often a binary matrix (consisting of 0s and 1s) that indicates whether there is an edge between two vertices.

   Now, the term "bit matrix" is more specific and refers to a matrix where each entry is a bit, which is a binary digit (0 or 1). In the context of adjacency matrices for graphs, the use of 0s and 1s to represent the absence or presence of edges between vertices leads to the association with the term "bit matrix."

   So, calling an adjacency matrix a "bit matrix" is more about emphasizing that the elements of the matrix are binary, representing whether edges exist (1) or do not exist (0) between the corresponding vertices in a graph. It's a way of highlighting the use of bits to encode graph connectivity information.