Q1. Linear Search + Deletion

Algorithms:

Step-1: Start

Step-2: Declare INTEGER ARRAY[MAX_SIZE], N, ITEM, FOUND := 0 [ARRAY for input, N for size, ITEM to search, FOUND flag]

Step-3: Prompt user to enter SIZE N and input ARRAY[0 to N-1] [Input values from user]

Step-4: Prompt user to enter ITEM to search [Get element to search]

Step-5: FOR I := 0 to N-1
IF ARRAY[I] == ITEM
SET FOUND := 1 [Item found flag set]
FOR J := I to N-2
ARRAY[J] := ARRAY[J+1] [Shift elements to delete]
DECREASE N by 1
BREAK

Step-6: IF FOUND == 0
DISPLAY "Search is unsuccessful" [Item not found message]
ELSE
DISPLAY "Item found and deleted" [Item found and deleted message]

Step-7: DISPLAY updated ARRAY

Step-8: End

Q2. Bubble Sort + Binary Search + Insert If Not Found

Algorithms:

Step-1: Start

Step-2: Declare INTEGER ARRAY[MAX_SIZE], N, ITEM, FOUND = 0 [ARRAY for input, N for size, ITEM to search]

Step-3: Prompt user to enter SIZE N and input ARRAY[0 to N-1] [Input values from user]

Step-4: FOR I = 0 to N-2
FOR J = 0 to N-I-2
IF ARRAY[J] > ARRAY[J+1]
SWAP ARRAY[J] and ARRAY[J+1] [Bubble Sort in ascending order]

Step-5: Prompt user to enter ITEM to search

Step-6: SET LOW = 0, HIGH = N-1

Step-7: WHILE LOW <= HIGH
MID = (LOW + HIGH) / 2
IF ARRAY[MID] == ITEM
SET FOUND = 1
BREAK
ELSE IF ITEM < ARRAY[MID]
HIGH = MID - 1
ELSE
LOW = MID + 1

Step-8: IF FOUND == 1
DISPLAY "Search is successful"
ELSE
INSERT ITEM at correct position to maintain sort [Find position and insert ITEM]
INCREASE N by 1

Step-9: DISPLAY updated ARRAY

Step-10: End

Q3. Linked List: Count + Insert at Specific Position

Algorithms:

Step-1: Start

Step-2: Define STRUCT NODE with INTEGER DATA and NODE *NEXT [Linked list node definition]

Step-3: Create HEAD pointer and initialize to NULL

Step-4: Create linked list by dynamically allocating nodes [User input loop to create list]

Step-5: Initialize COUNT = 0

Step-6: Traverse list using TEMP pointer
INCREMENT COUNT for each node [Count total nodes]

Step-7: Prompt user to enter POSITION and VALUE to insert

Step-8: Create NEWNODE with VALUE

Step-9: IF POSITION == 1
Set NEWNODE->NEXT = HEAD
HEAD = NEWNODE [Insert at beginning]
ELSE
Traverse to (POSITION - 1)th node
Insert NEWNODE between nodes

Step-10: DISPLAY total COUNT and updated list

Step-11: End

Q4. Search in Linked List and Delete if Found

Algorithms:

Step-1: Start

Step-2: Define STRUCT NODE with INTEGER DATA and NODE *NEXT

Step-3: Create HEAD pointer and initialize list

Step-4: Prompt user to enter ITEM to search and delete

Step-5: Initialize PREV = NULL, CURR = HEAD, FOUND = 0

Step-6: WHILE CURR is not NULL
IF CURR->DATA == ITEM
SET FOUND = 1
IF PREV == NULL
HEAD = CURR->NEXT [Delete head node]
ELSE
PREV->NEXT = CURR->NEXT [Delete middle or last node]
FREE CURR
BREAK
PREV = CURR
CURR = CURR->NEXT

Step-7: IF FOUND == 0
DISPLAY "Node not found"
ELSE
DISPLAY "Node deleted"

Step-8: DISPLAY updated linked list

Step-9: End

Q5. Tree: Create + Traversals + Search

Algorithms:

Step-1: Start

Step-2: Define STRUCT NODE with INTEGER DATA, NODE *LEFT, *RIGHT

Step-3: Create ROOT = NULL

Step-4: Insert nodes into tree by recursive function [User provides values one by one]

Step-5: Implement and call functions:
PREORDER(ROOT) [Root -> Left -> Right]
INORDER(ROOT) [Left -> Root -> Right]
POSTORDER(ROOT) [Left -> Right -> Root]

Step-6: Prompt user to enter ITEM to search

Step-7: Call SEARCH(ROOT, ITEM) recursively
IF found, return 1

Step-8: IF ITEM is found
DISPLAY "Search is successful"
ELSE
DISPLAY "Item isn’t found"

Step-9: End