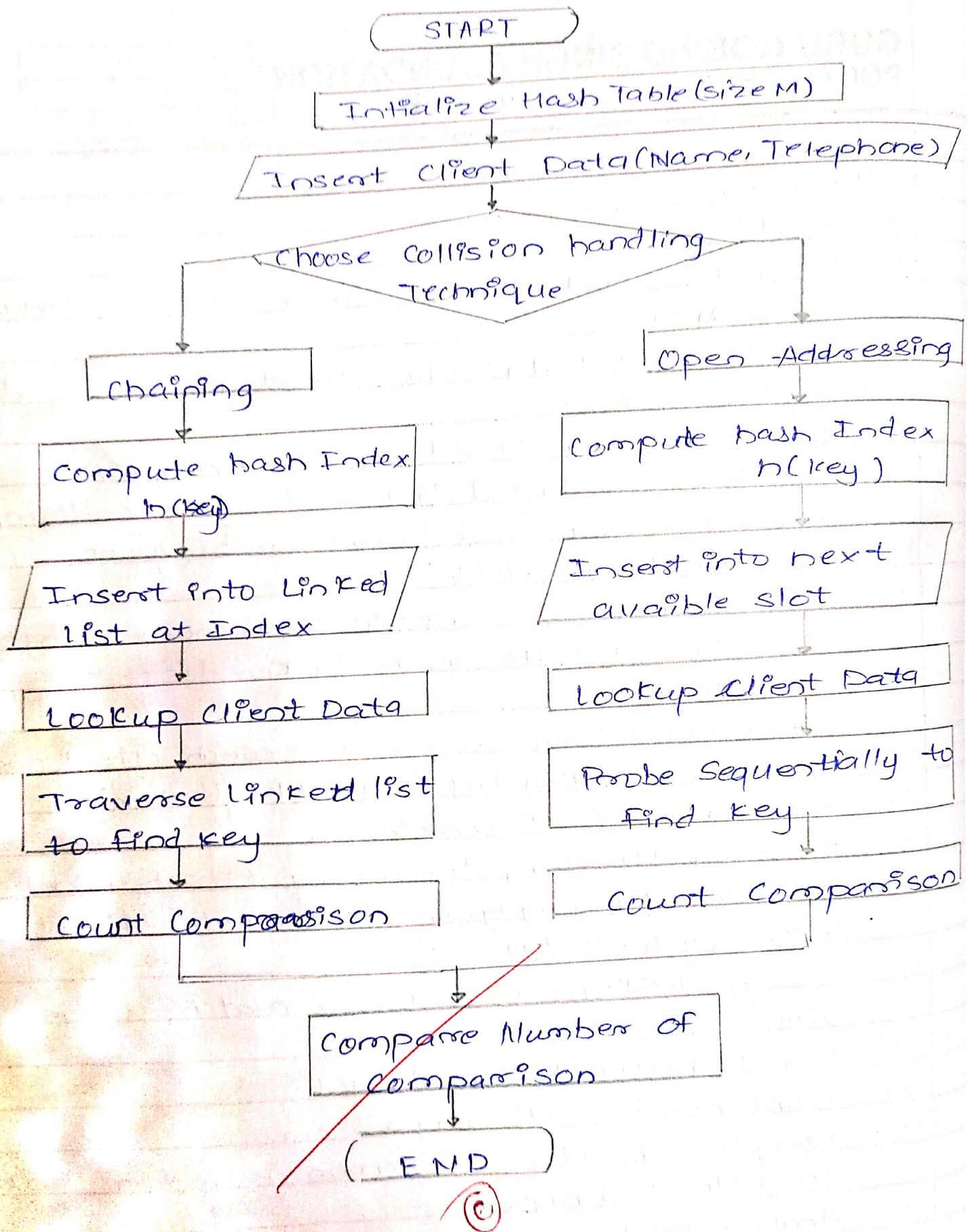


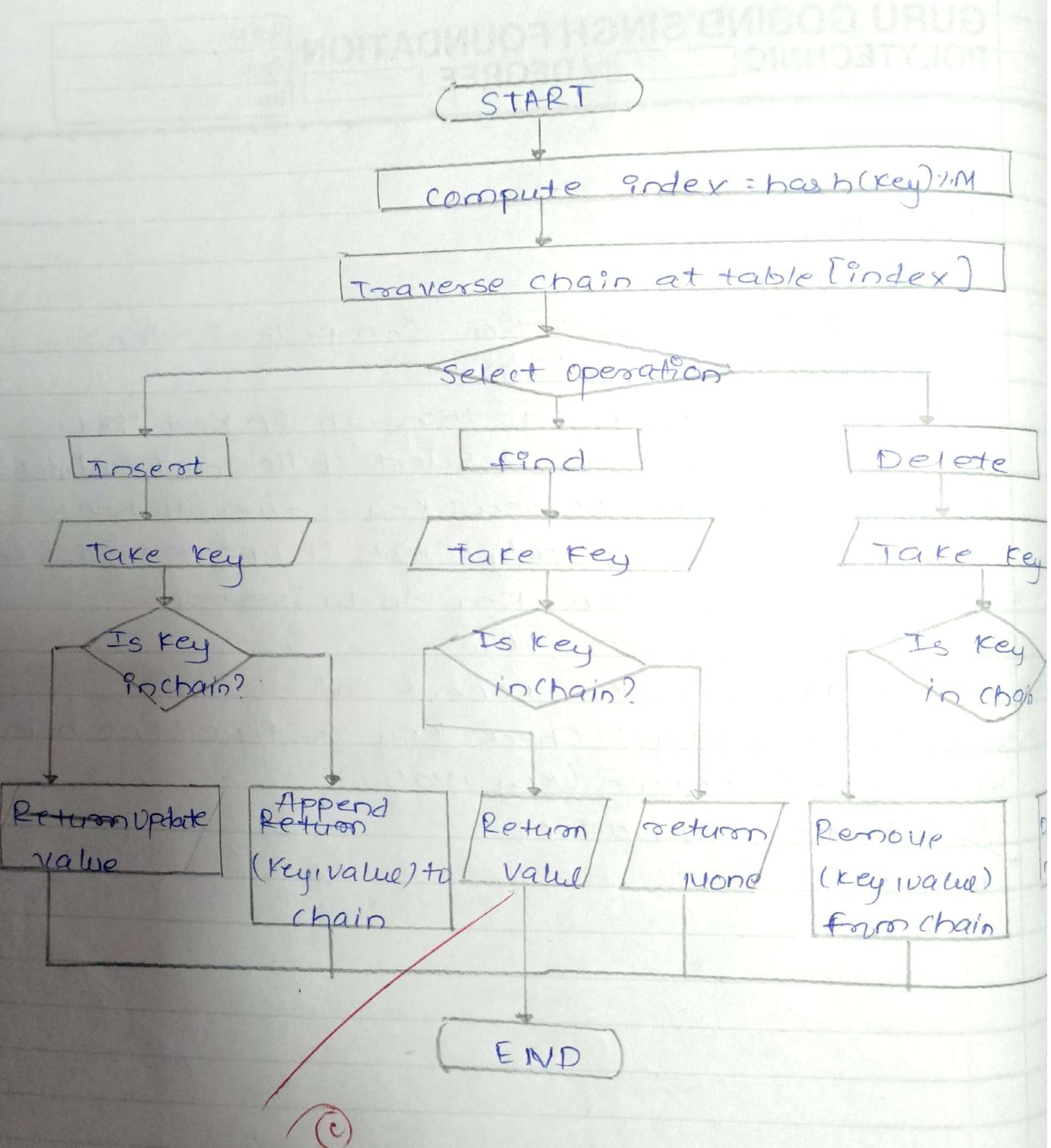
Algorithm:

1. Start
2. Initialize hash table : create a hash table of size ~~M~~ M
3. Insert Client Data : Add client name, and telephone number.
4. Choose Collision Handling Technique:  
 Chaining : Use linked list to handle collision  
 Open Addressing : Use linear probing to find next available slot.
5. If Chaining then, compute Hash index using  $h_1$
6. Then, Insert Data into linked list at Compute index.
7. Lookup Client Data : Process to search client num
8. Transverse linked list to find key(index) .
9. Count number of comparison.
10. Else : Open Addressing then, compute hash index  $h_1$
11. If Index slot empty, Insert key value . if not  
 $\text{index} = (\text{index} + 1) \% M$
12. Search (Lookup) key at index match , return value.
13. If not then probe next slot.
14. Count number of comparisons.
15. Compare number of comparison for lookup in both technique
16. Analyze performance.
17. STOP



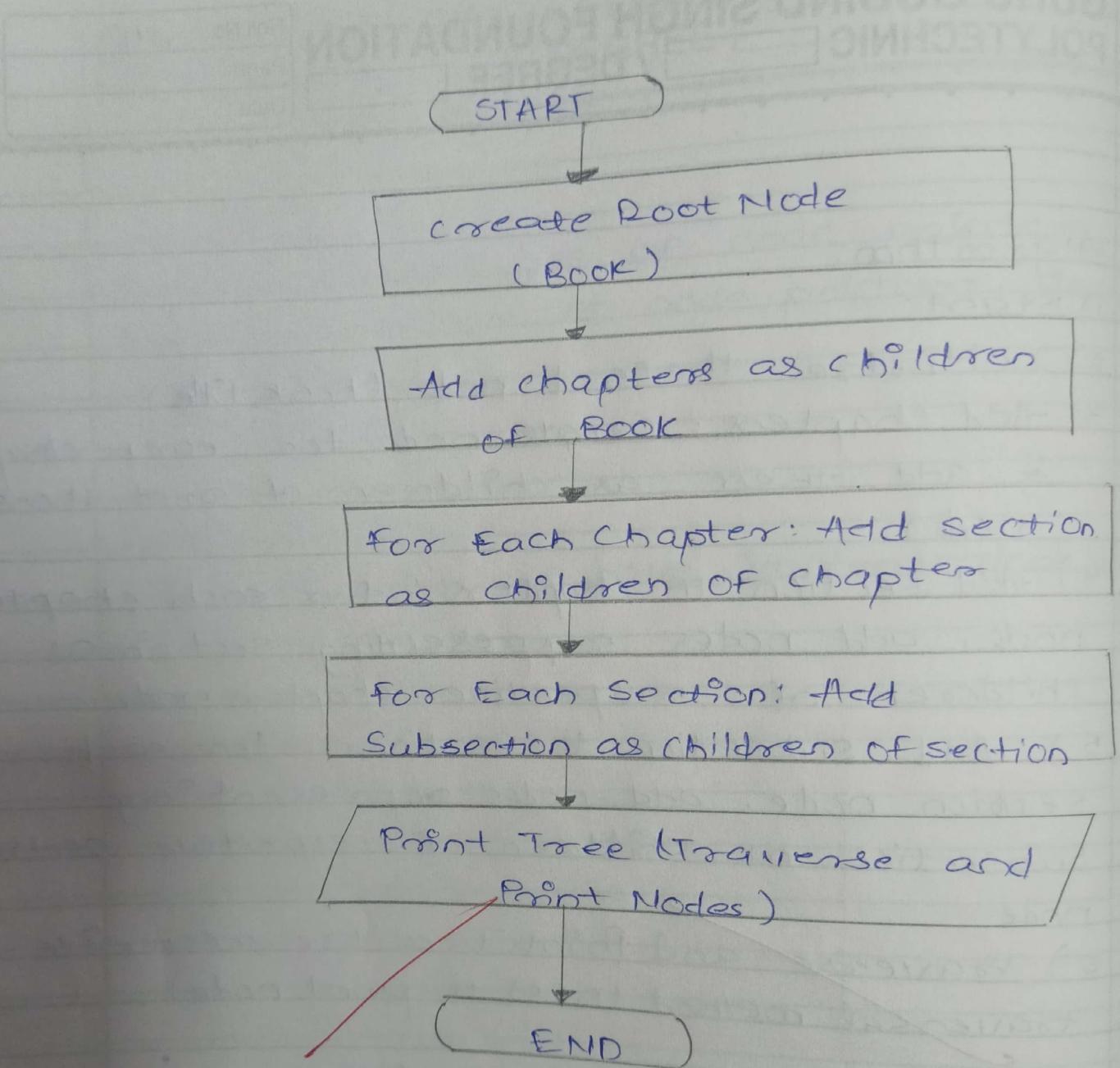
### Algorithm:

- 1) Start
2. Hash Treat Function: Compute index:  
$$\text{index} = \text{hash}(\text{key}) \% N$$
3. Traverse chain: go through linked list  
at computed index & select choice Insert, Find, Delete
4. If Insert(key): If add new(key, value) to hash  
table. If key already exist it update value & end
5. If Find(key): + search Key to be search.
6. If Yes: Return value
7. If No: Return none. & end
8. If Delete(key): Check key is found or not.
9. If Yes: Remove (key, value)
10. If No: Do nothing.
11. End.



Algorithm:

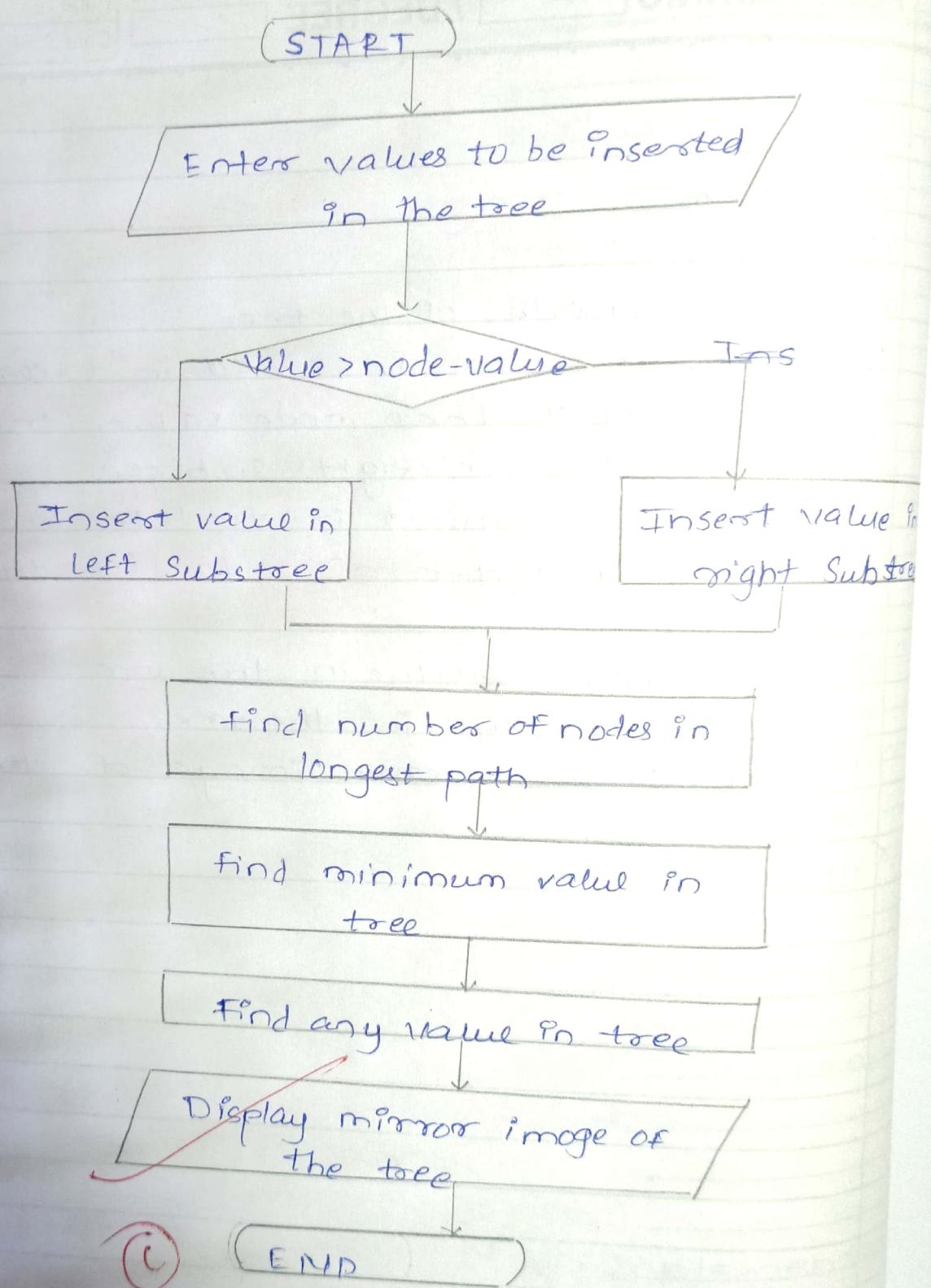
- 1) Start.
- 2) Initialize the root node (book title).
- 3) Add chapters: create node for each chapter & add them as children of root (book node).
- 4) Add section to chapters: for each chapter node, add nodes representing section as children of respective chapter node.
- 5) Add subsection to sections: for each section node, add nodes representing subsections as children of respective section node.
- 6) ~~Traverse and Print Tree: use recursive traversal method (DFS) to print node~~
- 7) End: trees built and printed.



©

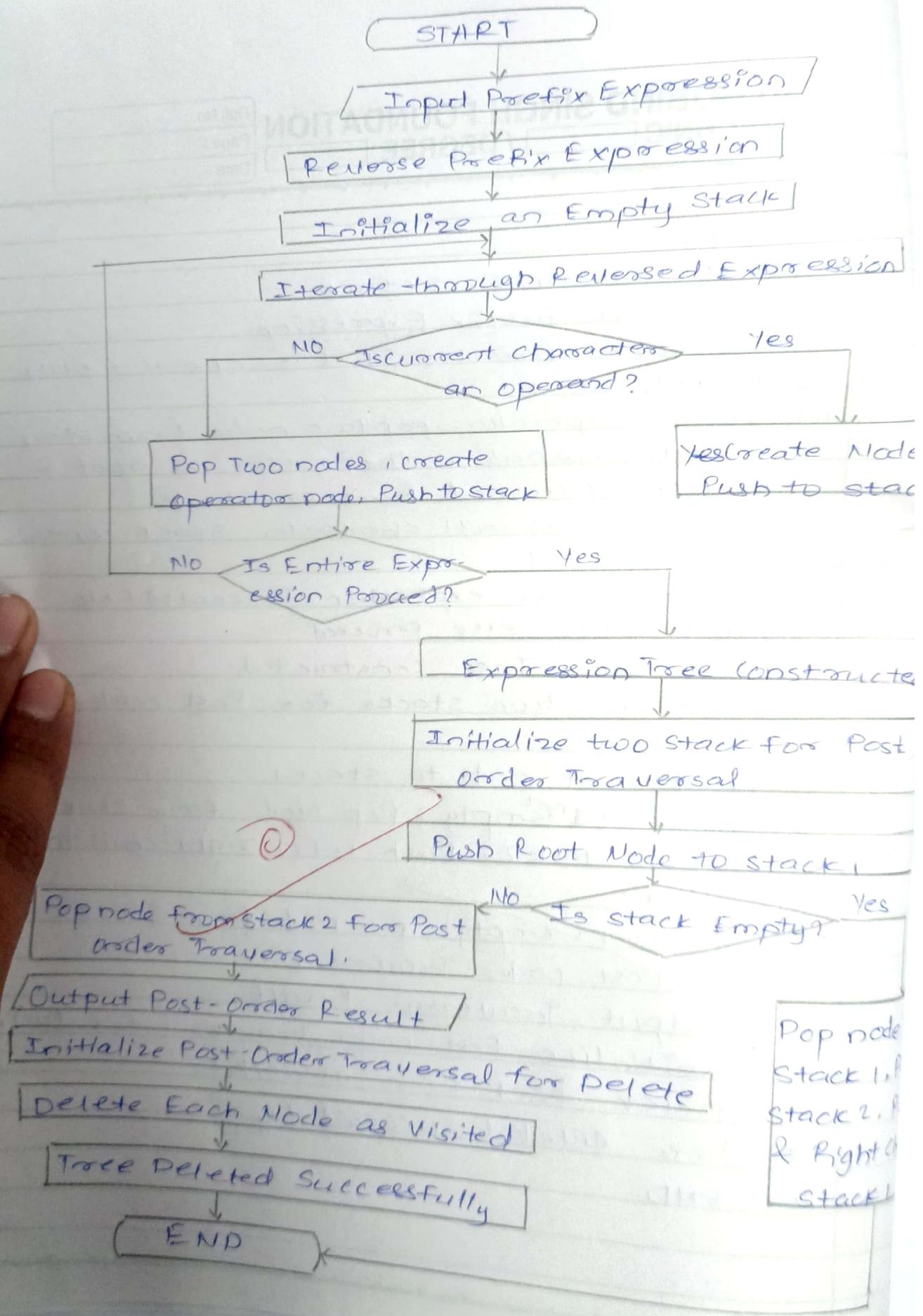
### Algorithm

- 1) Start
- 2) Create node of the tree
- 3) Enter values to be inserted in tree.
- 4) If value greater than node value then  
    insert value to right subtree
- 5) If not then insert it to left subtree
- 6) Find the number of nodes in the longest path.
- 7) Find minimum value in the tree.
- 8) Find any value in the tree.
- 9) Display the mirror image of the tree.
- 10) Stop



### Algorithm:

- 1) Start : Initialize a stack.
- 2) Take Input Prefix Expression.
- 3) For each character if it is operand push it onto stack.
- 4) If it is operator, pop two nodes from stack. Create new node with operator as root & push into new node back on stack.
- 5) After process all character stack contains only root node.
- 6) Check Entire expression proceed ? IF NO : go to step 5 else proceed.
- 7) Expression Tree constructed.
- 8) Initialize two stacks for Post-order Traversal.
- 9) Push root node to stack 1
- 10) IF stack 1 <sup>not</sup> empty : Pop Node from Stack 1. Push to Stack 2, Push left, right child to stack 1
- 11) IF stack 1 empty : pop node from stack 2   
~~for Post order Traversal.~~
- 12) Output Traversal Result.
- 13) Initialize Post-order Traversal for Delete.
- 14) Delete each node as visited.
- 15) Tree deleted successfully.
- 16) END.

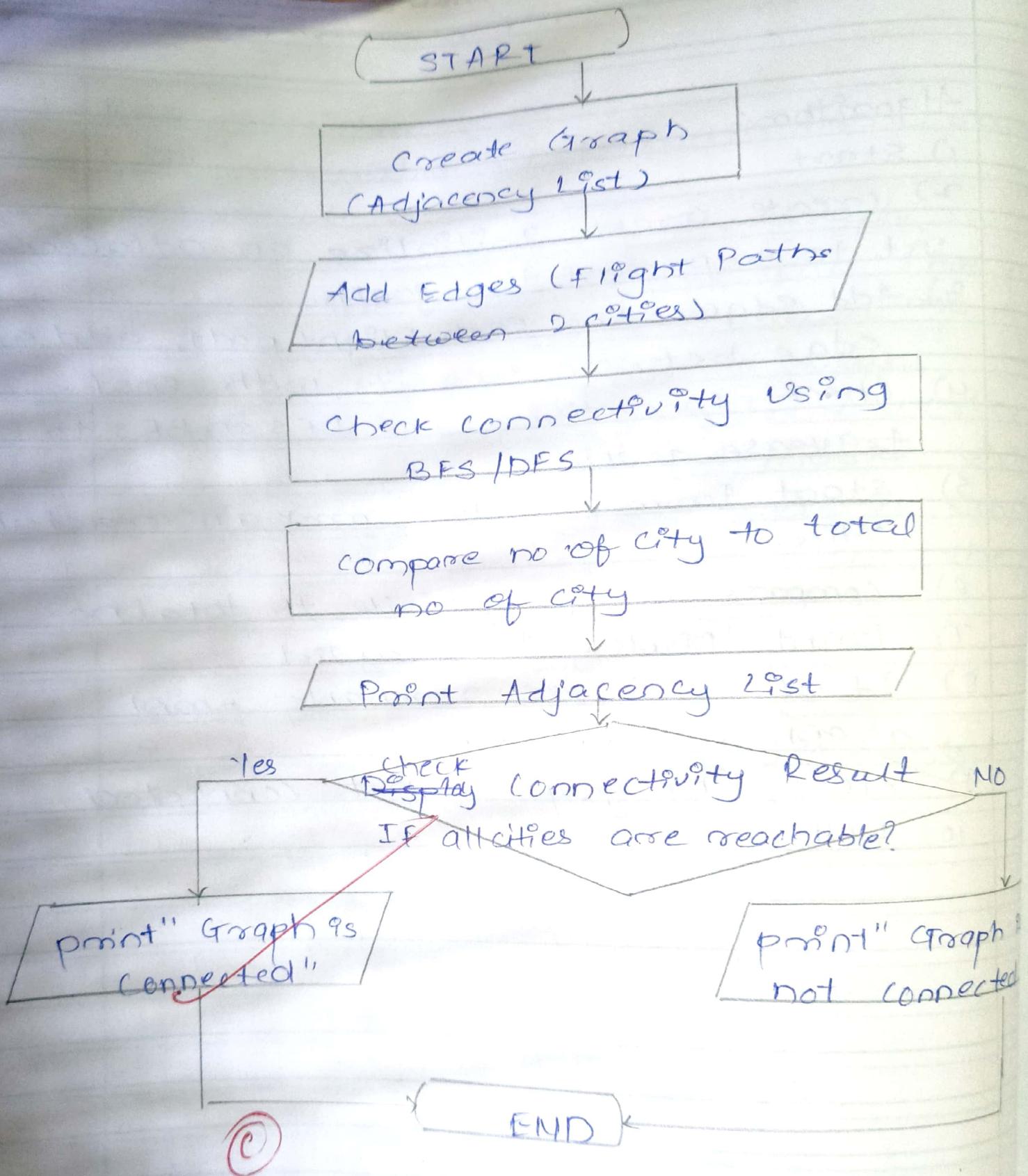


# GURU GOBIND SINGH FOUNDATION POLYTECHNIC [ ] / DEGREE [ ]

Roll No.:	64
Page:	
Date:	

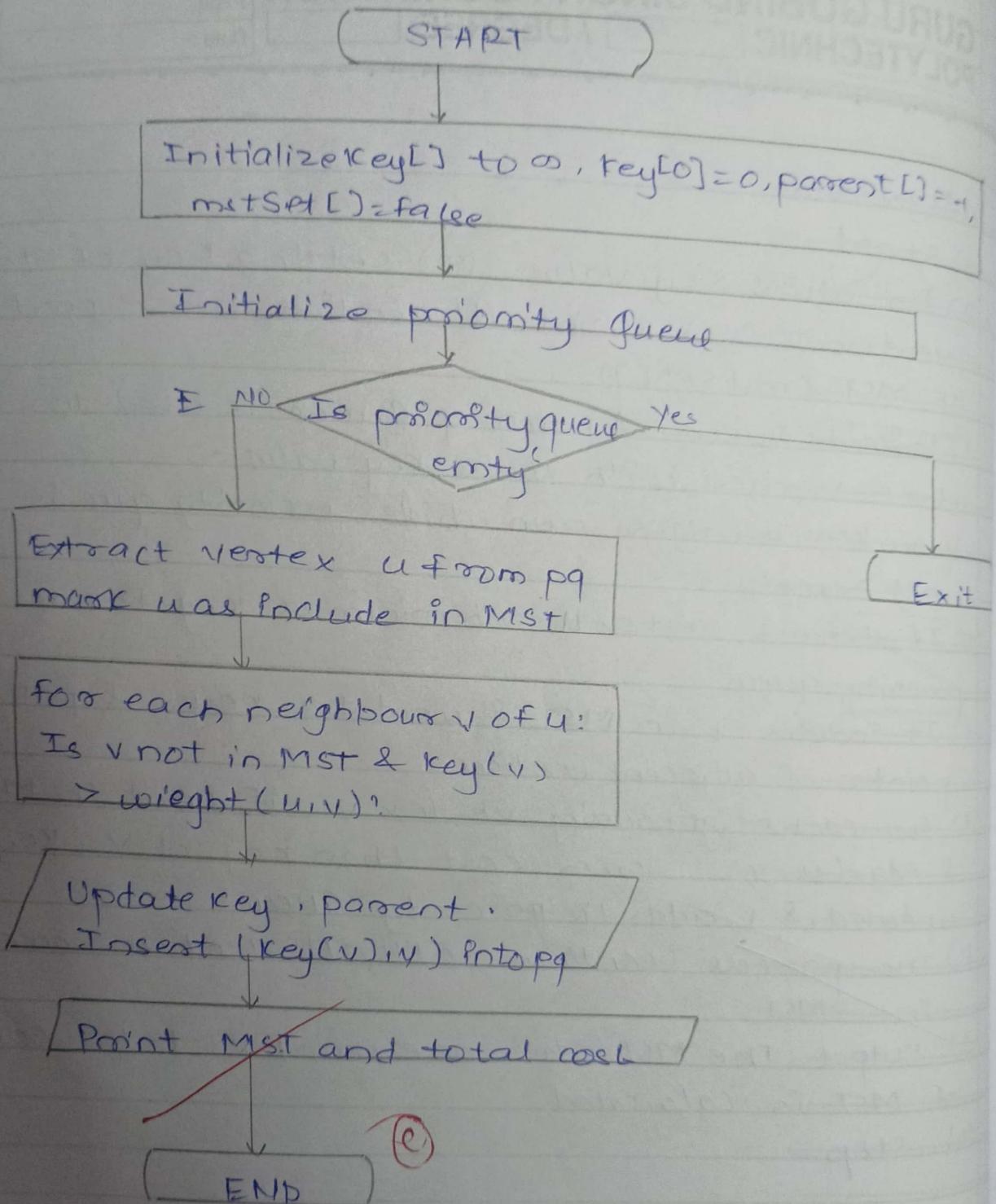
Algorithm:

- 1) Start
- 2) Create graph: Initialize an adjacency list to represent the graph.
- 3) Add edges: for each flight path, add an edge between two cities with cost.
- 4) Check connectivity: use DFS or BFS to traverse graph
- 5) Start from any city & mark all reachable cities.
- 6) Compare no. of city visited to total no.
- 7) Print / Display adjacency list.
- 8) If all cities are reachable print ~~graph is connected.~~
- 9) Else: The graph is not connected
- 10) END.



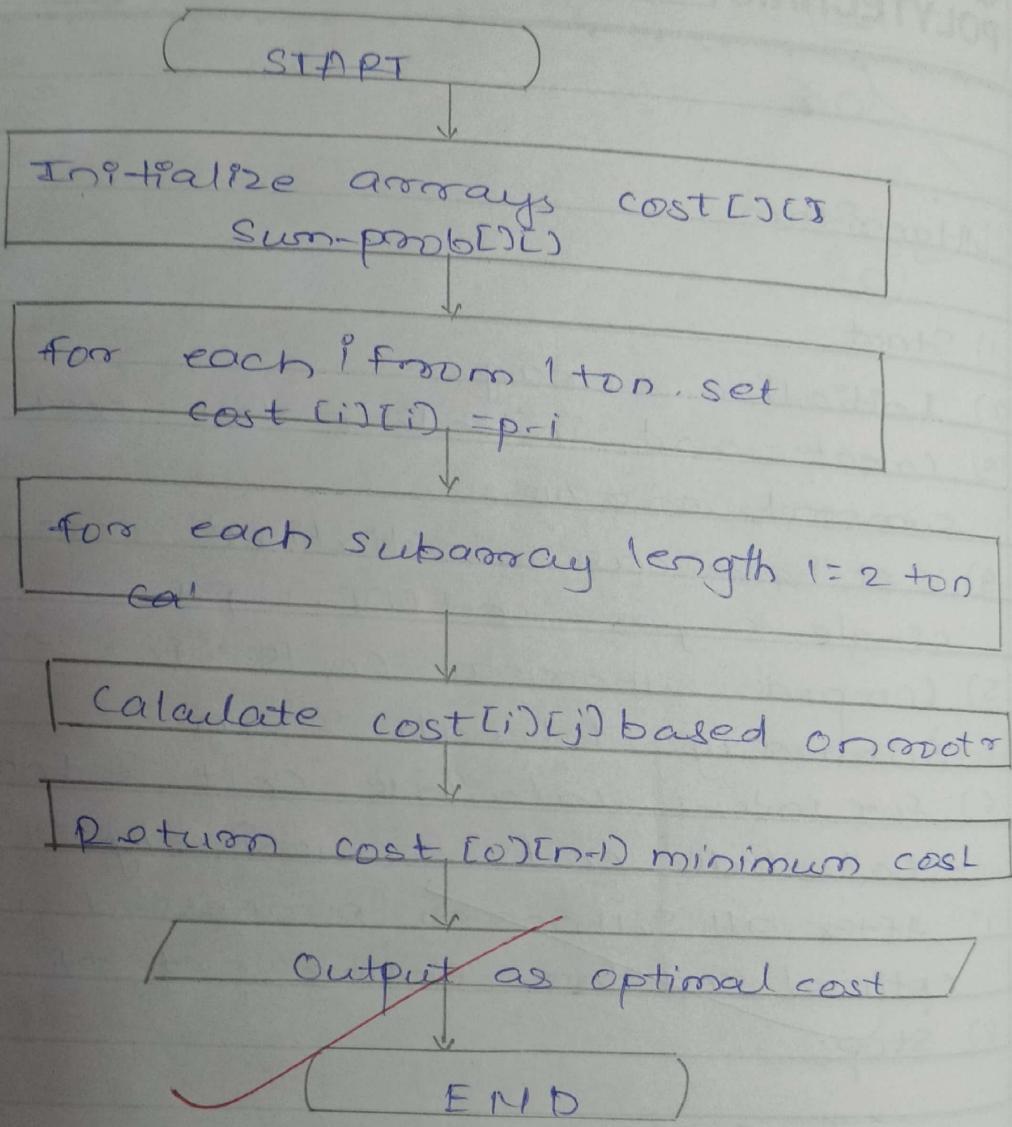
### Algorithm :

- 1) Start
- 2) Initialize key value ( $\infty$ ), setting key of start vertex to 0, & setting all vertices as not part of MST (mst set [ ])
- 3) Priority Queue : A priority queue is used to store vertices with their key value.
- 4) Main loop: The algorithm runs while priority que is not empty
- 5) If extract vertex u with minimum key from priority queue
- 6) It includes u in MST & updates keys & parent pointer of adjacent vertices
- 7) for each neighbouring vertex v, if v not in MST & edge(u,v) lower cost than key of v, key updated, & v adds to priority queue
- 8) The process continues until all vertices are in MST
- 9) Output : The MST edges are print, & total cost of MST is calculated
- 10) Stop



### Algorithm:

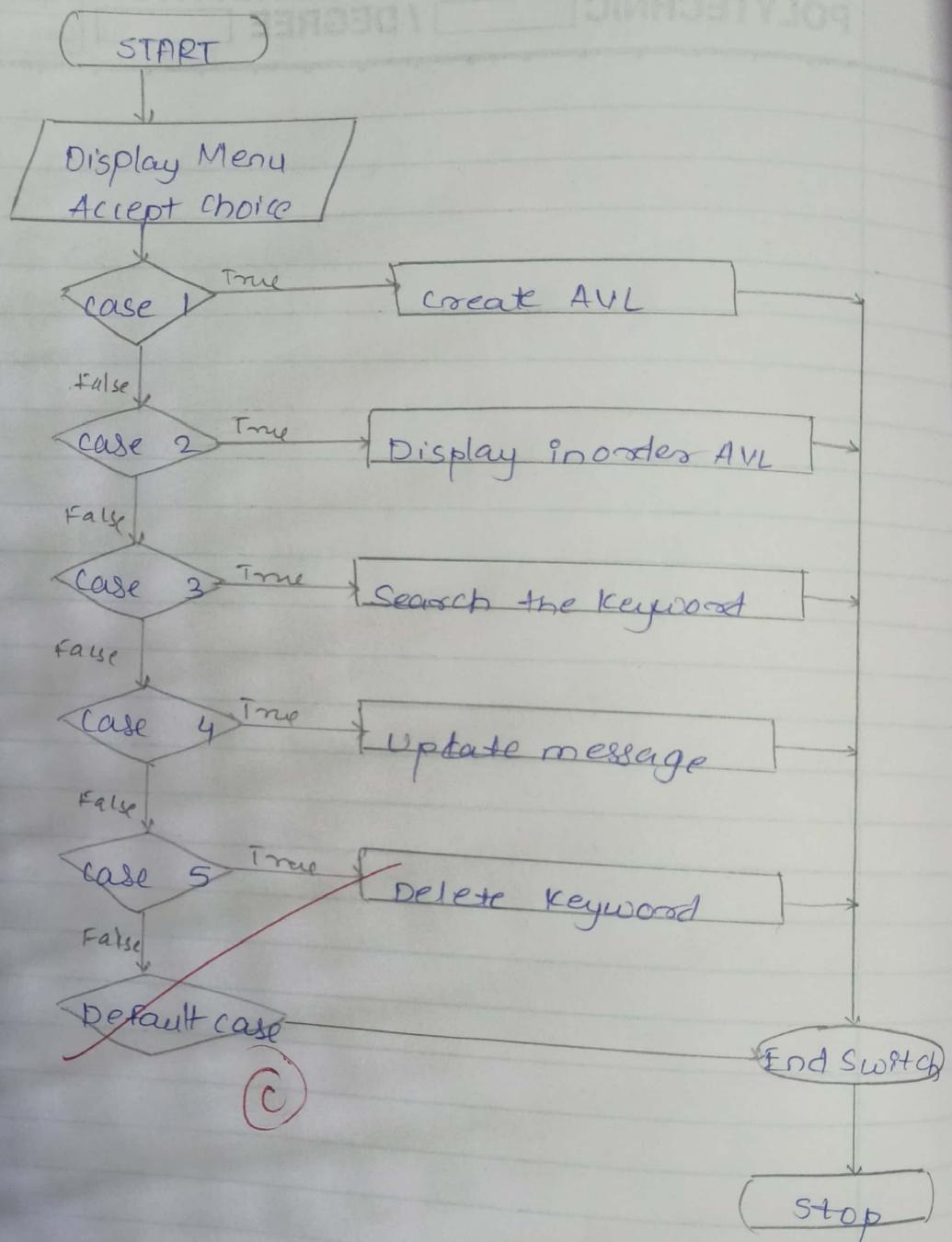
- 1) Start
- 2) Initialize arrays.
- 3) Create and initialize the cost and summb. arrays
- 4) Set Base case: Initialize cost for a single key as  $\text{cost}[i][i] = p_i$
- 5) Compute subarray costs for each subarray of length 2 to n
- 6) For index i, calculate cost & for root k, calculate cost recursively.
- 7) After all subarray processed, give optimal cost for entire set of keys
- 8) Stop.



(C)

Algorithm:

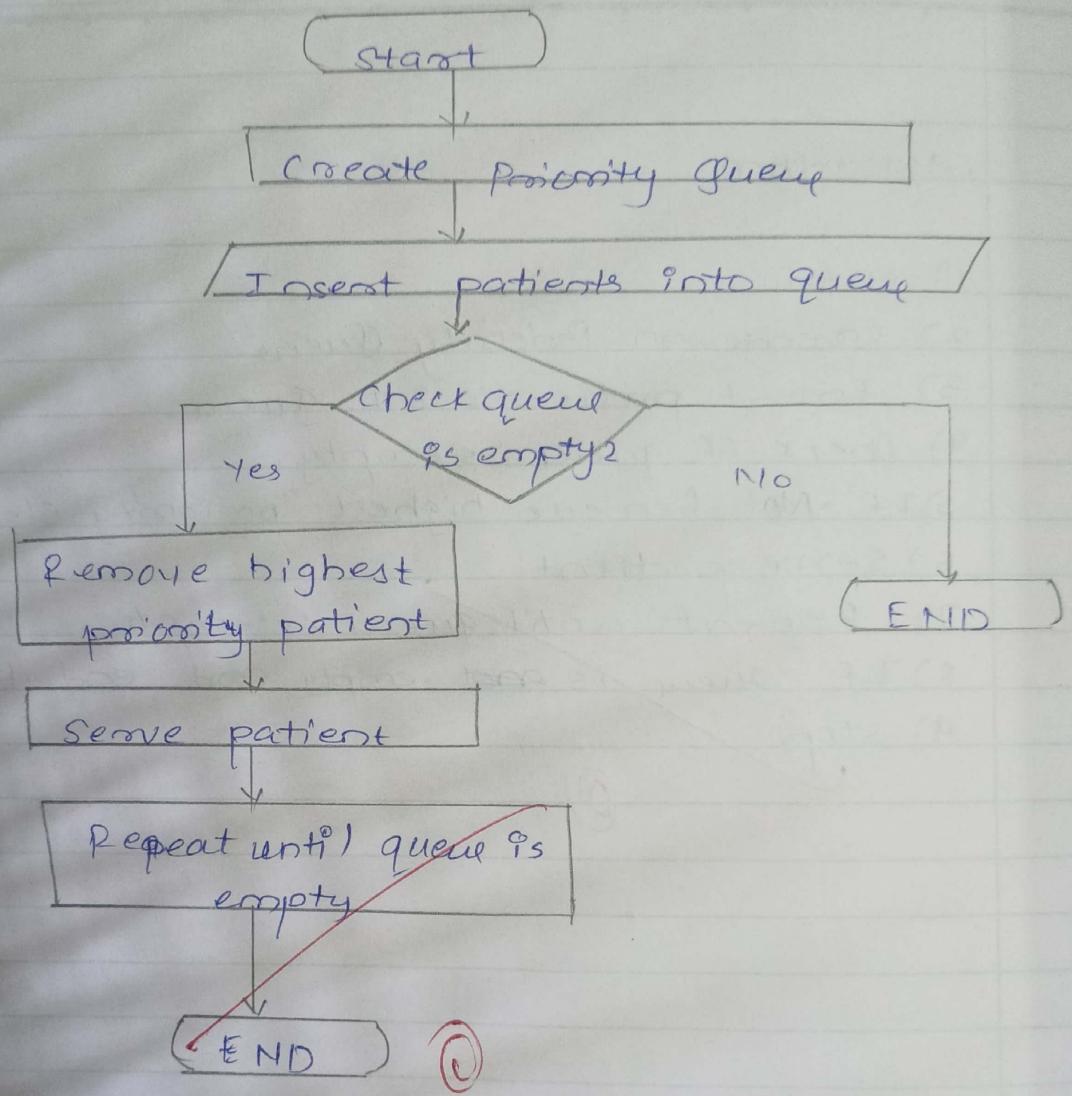
- 1) start.
- 2) Insert Keyword: Search if keyword already exist, if not insert and rebalance the tree.
- 3) Delete Keyword: Search for the keyword and remove it, then rebalance
- 4) Update Keyword: Search for keyword, update the meaning.
- 5) Display Sorted: Traverse tree in-order for ascending or reverse in-order for descending
- 6) Search Keyword: Traverse tree to find the keyword.
- 7) find minimum comparisons: measure maximum depth of tree for worst-case search
- 8) END



Algorithm:

- 1) Start
- 2) Create an Priority Queue.
- 3) Insert patients into queue.
- 4) Check if queue is empty.
- 5) IF NO: Remove highest priority element
- 6) Serve patient
- 7) Repeat until queue is empty
- 8) IF queue is ~~not~~ empty and end process
- 9) STOP

①



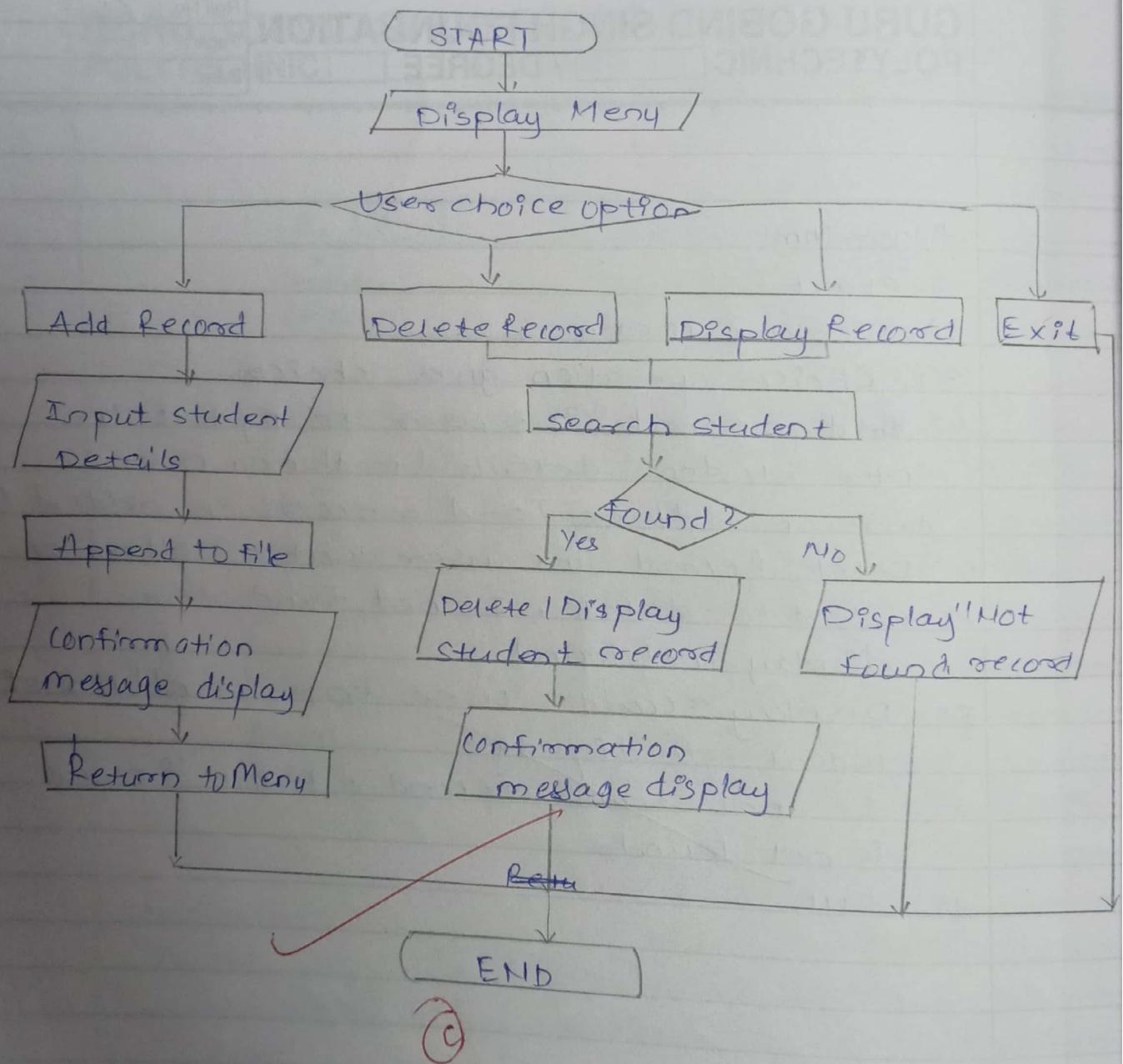
# GURU GOBIND SINGH FOUNDATION POLYTECHNIC

/ DEGREE

Roll No.:	64
Page:	
Date:	

## Algorithm:

- 1) Start
- 2) Display menu
- 3) Choice operation gives choices.
- 4) Add records: The user prompted to enter student details (roll no, name, division, address) and record in added file.
- 5) Delete Record: The user enters roll no of student to delete, If not found then display not found.
- 6) Display record: user enters roll no of student to display.
- 7) IF not found record, print "Record is not found"
- 8) END



# GURU GOBIND SINGH FOUNDATION POLYTECHNIC

/ DEGREE

Roll No. : 64

Page :

Date :

## Algorithm:

- 1) Start
- 2) Input employee details (ID, name, salary, designation)
- 3) Check employee ID exists in file or not
- 4) If Yes, then display "ID exists".
- 5) If No, Append employee to data file
- 6) Create and append index entry in index file
- 7) Sort the index file
- 8) Close the file
- 10) END

