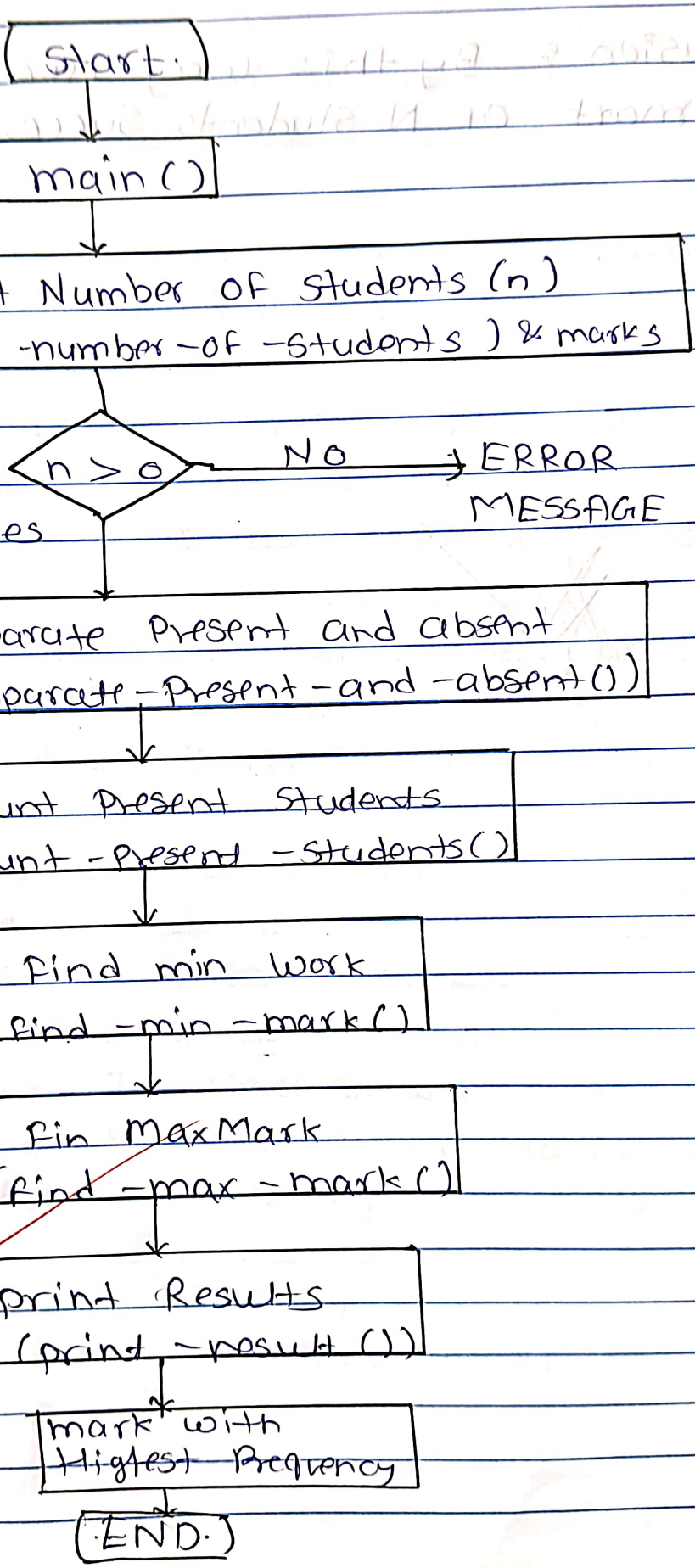
Algorithm :

1. Input the no. of students N.
2. Create an empty list students.
3. For each students from 1 to N
   - Input the students Name.
   - Input the students mark.
   - Store the students name & marks as a in the students list.
4. output the stores lists of students & the marks.

1-A-3

Flowchart:

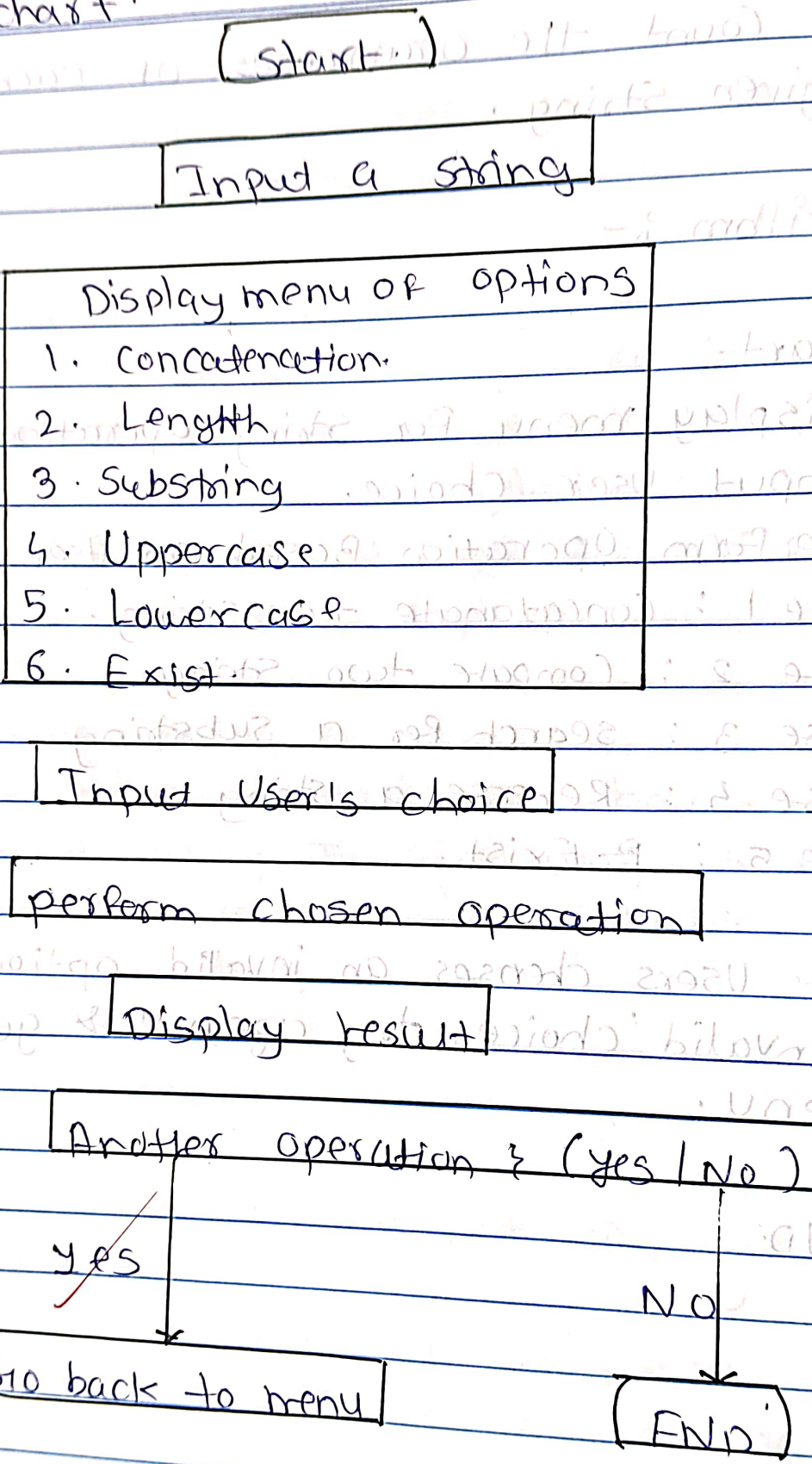


e] To Count the occurrences of each word in a given String.

Algorithm :-

1. Start
2. Display menu for string operation.
3. Input user choice.
4. Perform Operation Based on User choice :
   Case 1 : Concatanate two string.
   Case 2 : Compare two string.
   Case 3 : search for a substring
   Case 4 : Reverse a string
   Case 5 : Exist.
5. If Users chooses an invalid option, display "Invalid choice, try again" & go back to menu.
6. END.

Flowchart:



Conclusion :-

By this way, we can perform string operations successfully.

Theory :-
Concept of saddle point in ...
example.

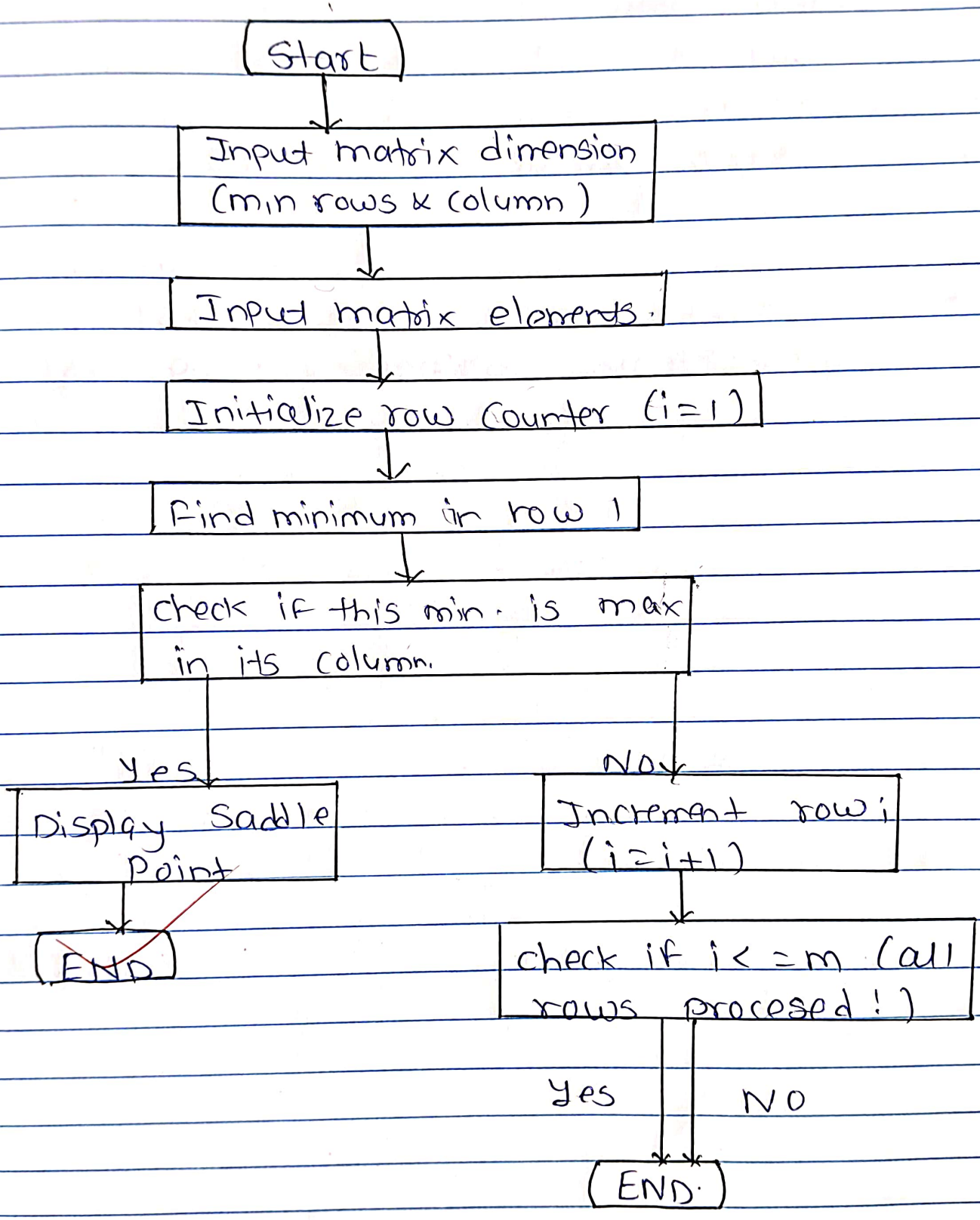
A Saddle point in a matrix is an element ... is the smallest in its row but the larg... in its column (or vice versa). In other wo... its a point that is both a local minimum ... its row & a local maximum in its column.

e.g.
$$\begin{bmatrix} 1 & 3 & 4 \\ 5 & 2 & 9 \\ 8 & 1 & 6 \end{bmatrix}$$

In second row, the element 2 is the small... In second column, 2 is the largest.
Thus, 2 is a saddle point

Algorithm :-
1. Input : A matrix A of size m×n
2. For each row :
   - Find the minimum element in the current ...
   - Identify the column index of this minim... element.
3. check the column : For the identified colum... check if this element is the largest elemen... in the column.

Flow chart :-


Start
Input matrix dimension (m,n rows & column)
Input matrix elements.
Initialize row counter (i=1)
Find minimum in row i
check if this min. is max in its column.
Yes
Display Saddle Point
END
NO
Increment row i (i=i+1)
check if i<=m (all rows processed!)
yes
NO
END.

4. If yes, this element is saddle point.
5. Repeat this process for all rows
6. Output the saddle point if it exists ; otherwise indicate that no saddle point is found.

Conclusion :-

By this way, we can determine location of saddle point in matrix successfully.

2. matrix Substraction.

$$A = \begin{bmatrix} 5 & 7 \\ 4 & 6 \end{bmatrix} \qquad B = \begin{bmatrix} 3 & 1 \\ 2 & 5 \end{bmatrix}$$

$$A-B = \begin{bmatrix} 5-3 & 7-1 \\ 4-2 & 6-5 \end{bmatrix} = \begin{bmatrix} 2 & 6 \\ 2 & 1 \end{bmatrix}$$

3. matrix multiplication.

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \qquad B = \begin{bmatrix} 3 & 4 \\ 5 & 6 \end{bmatrix}$$

$$A * B = \begin{bmatrix} 1*3+2*5 & 1\cdot 4+2\cdot 6 \\ 3\cdot 3+4\cdot 5 & 3\cdot 4+4\cdot 6 \end{bmatrix} = \begin{bmatrix} 13 & 16 \\ 29 & 36 \end{bmatrix}$$

4. Transpose.

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

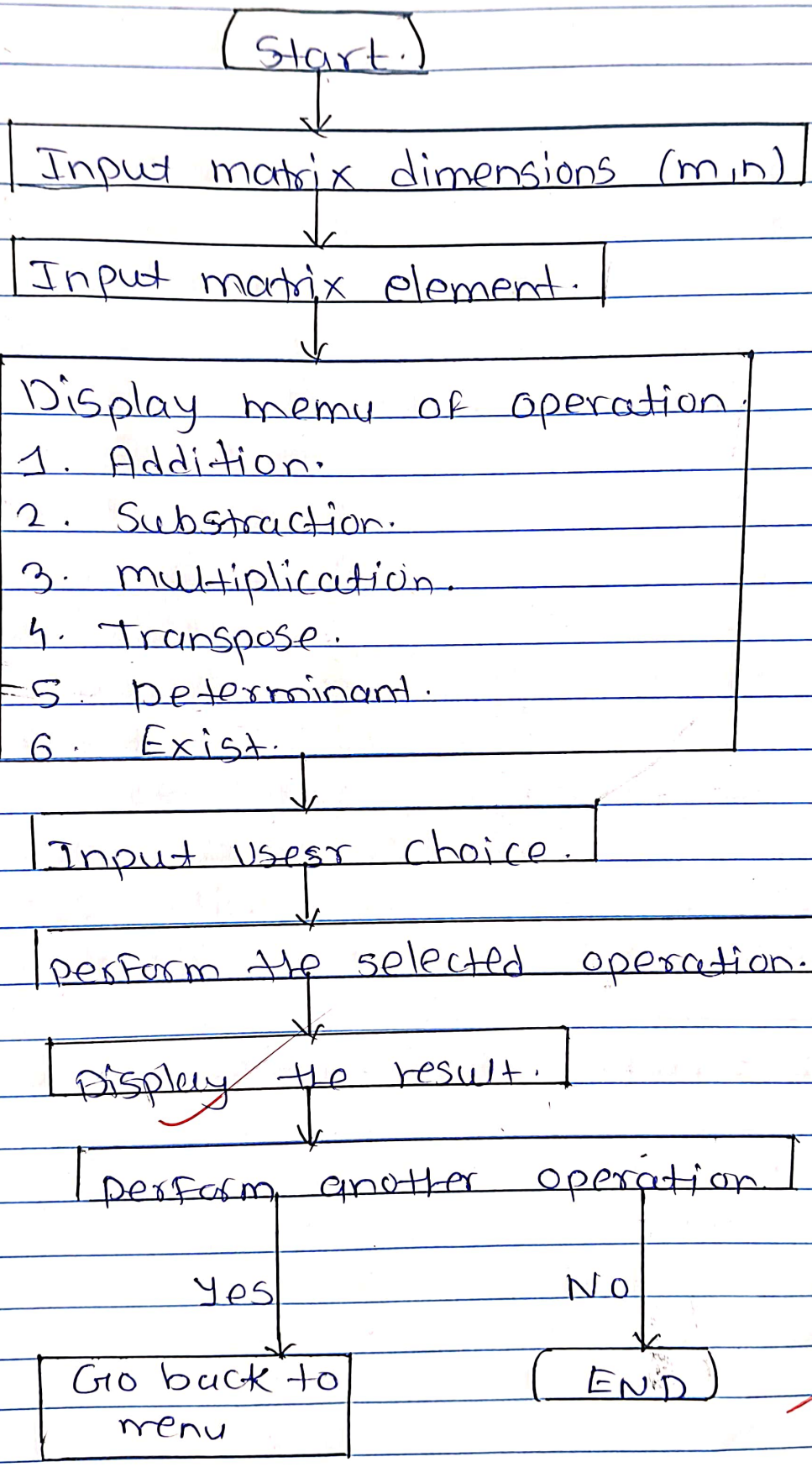
Transpose of $A = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$

Algorithm :-

1. Start.
2. Input number of rows & columns of first matrix.

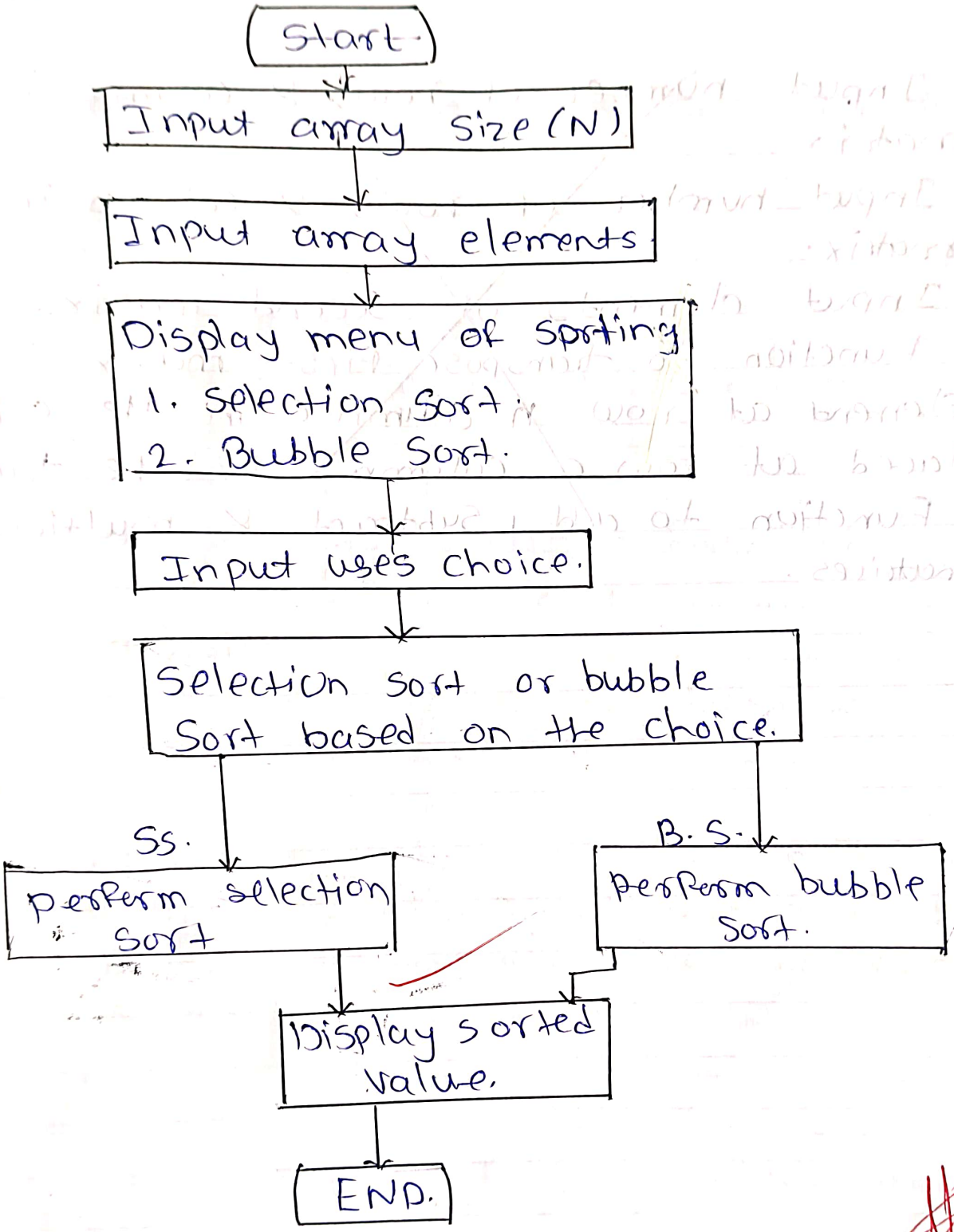
3. Input number of rows & column in second matrix.
4. Input number of rows & column in Second matrix
5. Input elements of second matrix.
6. Function to transpose first matrix i.e. the element at row n column c in the original is placed at row c column r of the transpose.
7. Function to add, Substract & multiply two matrices

Flowchart :-



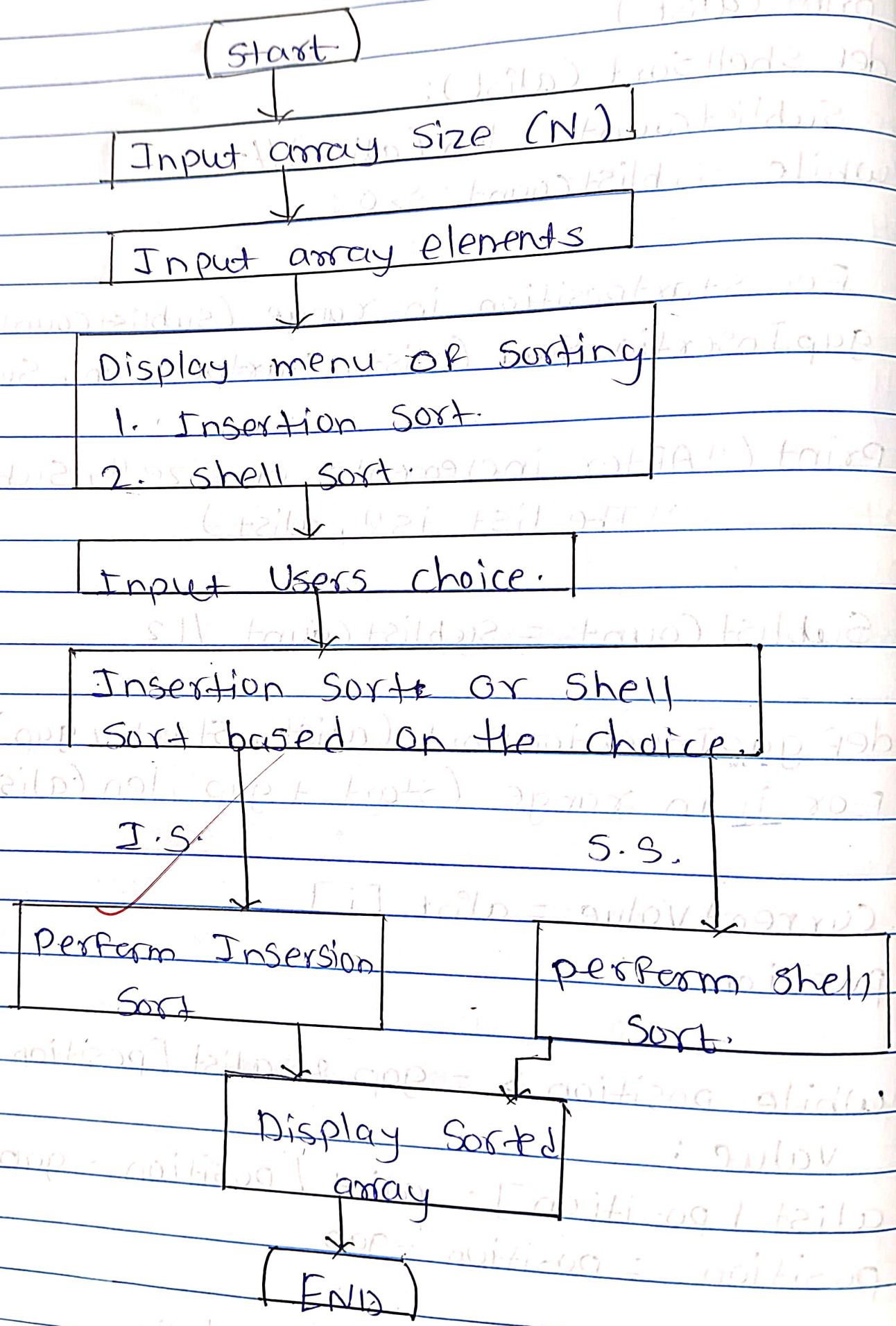
Conclusion :- By this way, we can perform various operations on matrix successfully.

pchart :-



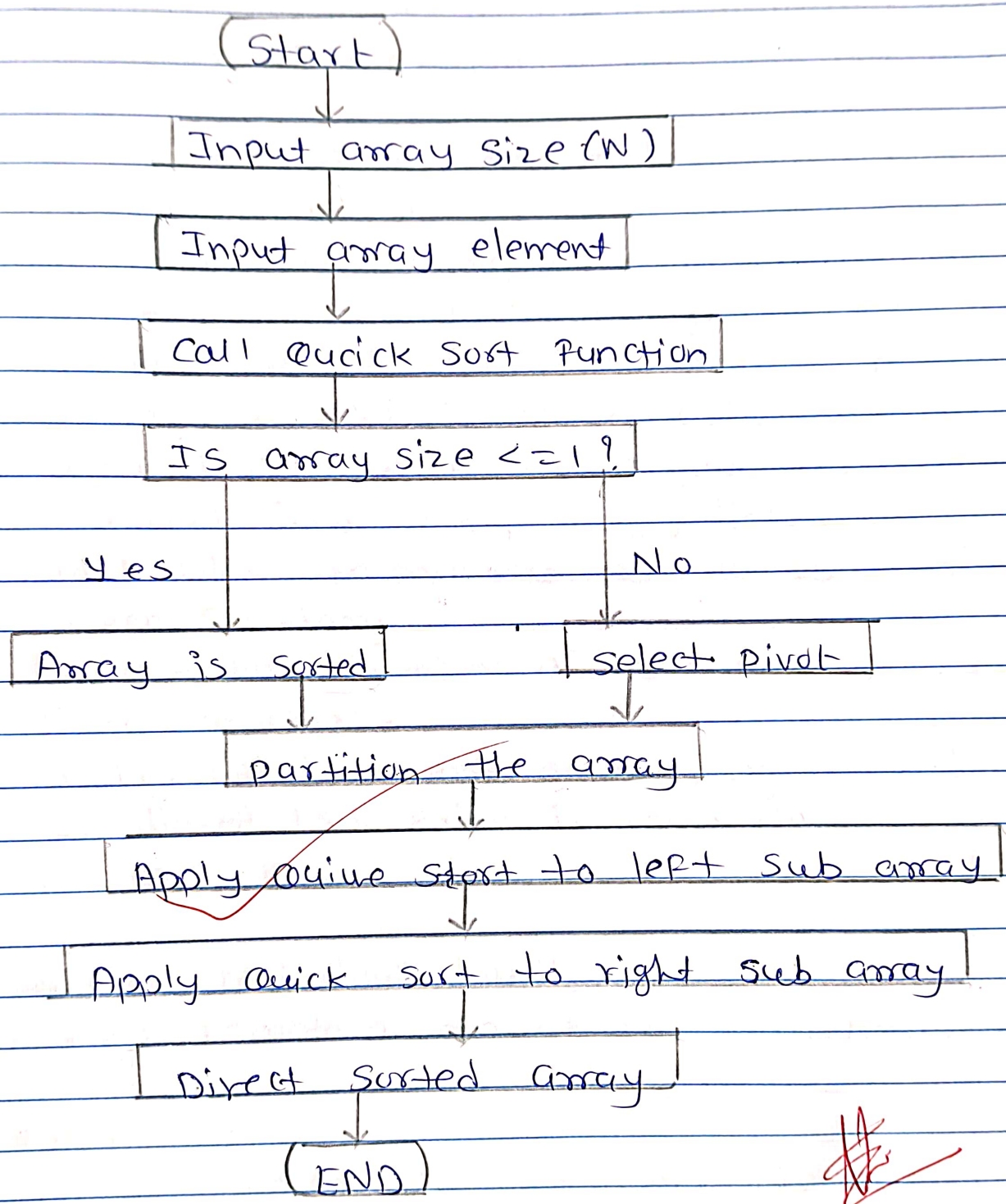
Conclusion :- By this way we can perform sorting of an array using selection & Bubble Sort.

Flowchart :-

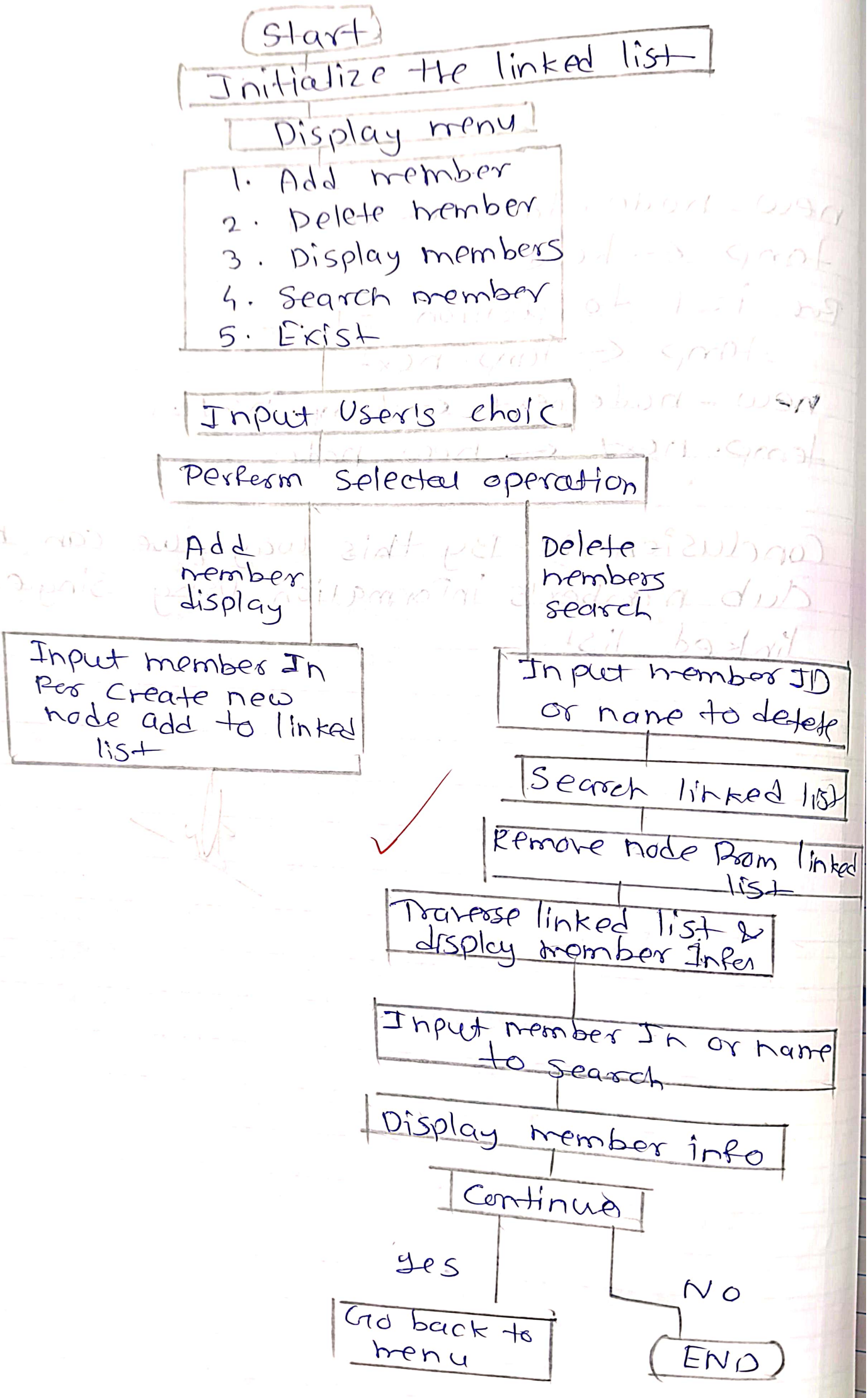

Start
Input array size (N)
Input array elements
Display menu of sorting
1. Insertion sort
2. shell sort
Input Users choice.
Insertion sort or Shell sort based on the choice.
I.S.
S.S.
Perform Insersion Sort
perform shell sort.
Display Sorted array
END

7 B-(5)

Flow chart:



Conclusion :- By this way, we can perform Sorting array of floating point no. in ascending order using Quick Sort


Start
Initialize the linked list
Display menu
1. Add member
2. Delete member
3. Display members
4. Search member
5. Exist
Input User's choice
Perform selected operation
Add member display
Delete members search
Input member ID or name to delete
Search linked list
Remove node from linked list
Traverse linked list & display member info
Input member ID or name to search
Display member info
Continue
yes
No
Go back to menu
END

9 c (5)

c) Disadvantage of CDLL :-

- memory overhead : Each node requires extra memory to store pointers to the next one previous nodes.

- Complex Implementation : managing pointers during insertion & deletion can be more complex compared to singly linked list.

- Algorithm :-

1) Initialize Theater Rows :
   i) operate an arrays of head pointers to store the rows.
   ii) For each row :
   iii) call the functions to create a circular doubly linked list of 7 ~~years~~ seats.

2) Create Row Function.
   i) Create the 1st node (seat) & set it as the head.
   ii) Loop to create the remaining seats, linking each node to its previous & next node.

3) Display Available seats :-
   i) Traverse and circular list starting from the head
   ii) For each seat, check if it is available (not booked)

9c (4)

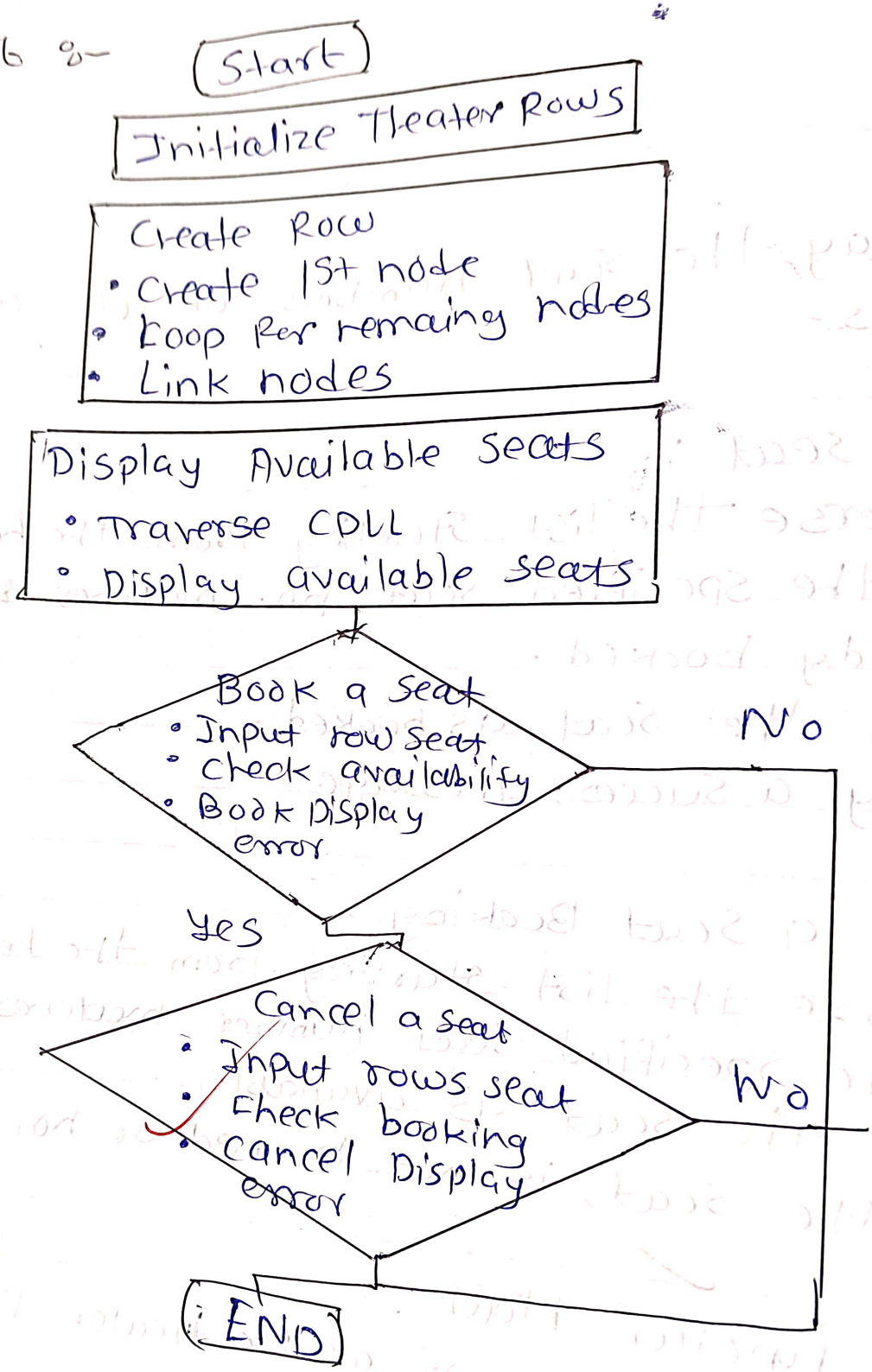
iii) Display the seat numbers of all available seats.

4) Book a seat :
i) Traverse the list starting from the head.
ii) If the specified seat no. matches & it not already booked.
- mark the seat as booked.
- Display a success message.

5) Cancel a seat Booking :
i) Traverse the list starting from the head.
ii) If the specified seat number matches & is booked
- mark the seat as available.
iii) If the seat is not booked or not found, display

6) main function Flow :
i) initialize the rows of the theater using the create row function.
ii) simulate booking & cancelation operation using the book & cancel function.
iii) Display the list of available seats before & after the operations.

Flowchart :-



Conclusion :-

By this way we can book or cancel movie ticket using doubly circular linked l

Input appointment ID or Details.

② Valid data Input.
- check if the movided User ID is valid.
- If not inform the User and request a valid User ID.
- check if the provided ID or details are not empty If empty, inform the User and promot For valid input.

Ensure the Format is Correct.
(e.g. Valid appointment ID)

* Algorithm :-

1) Node Structure Initialization.

- Create a structure (Appointment that include me start time, end time, booking status, and a pointer to the next code.

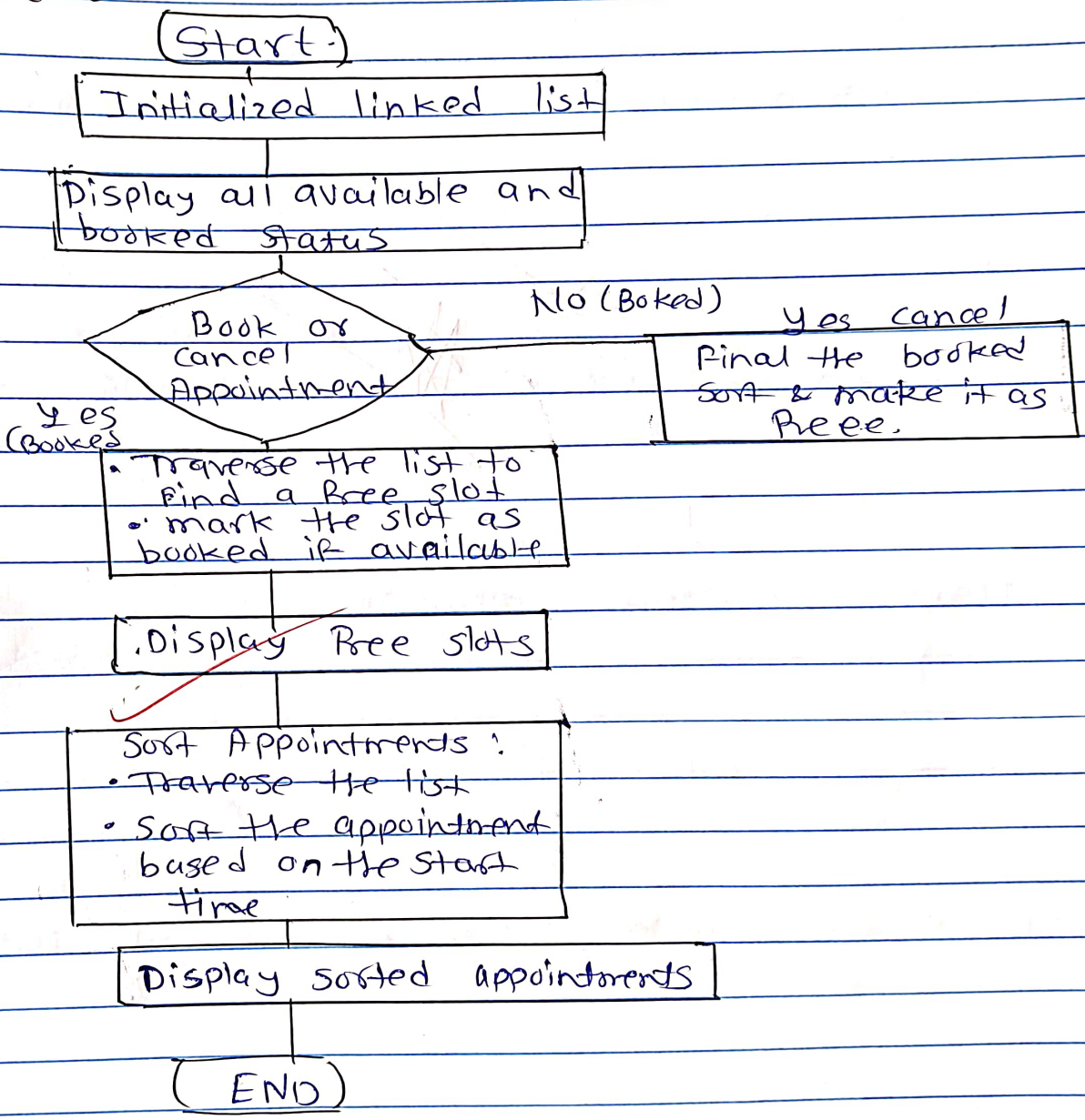
2) Creating Appointment Node
- create a function.

3) Creating Appointment Nod.
5) Creating Appointment Node :-
- create a function to initialize a new node with start and end times end set the booking status.

8) END

The process ends after displaying the sorted list of Uppointments.

Flow chart :-



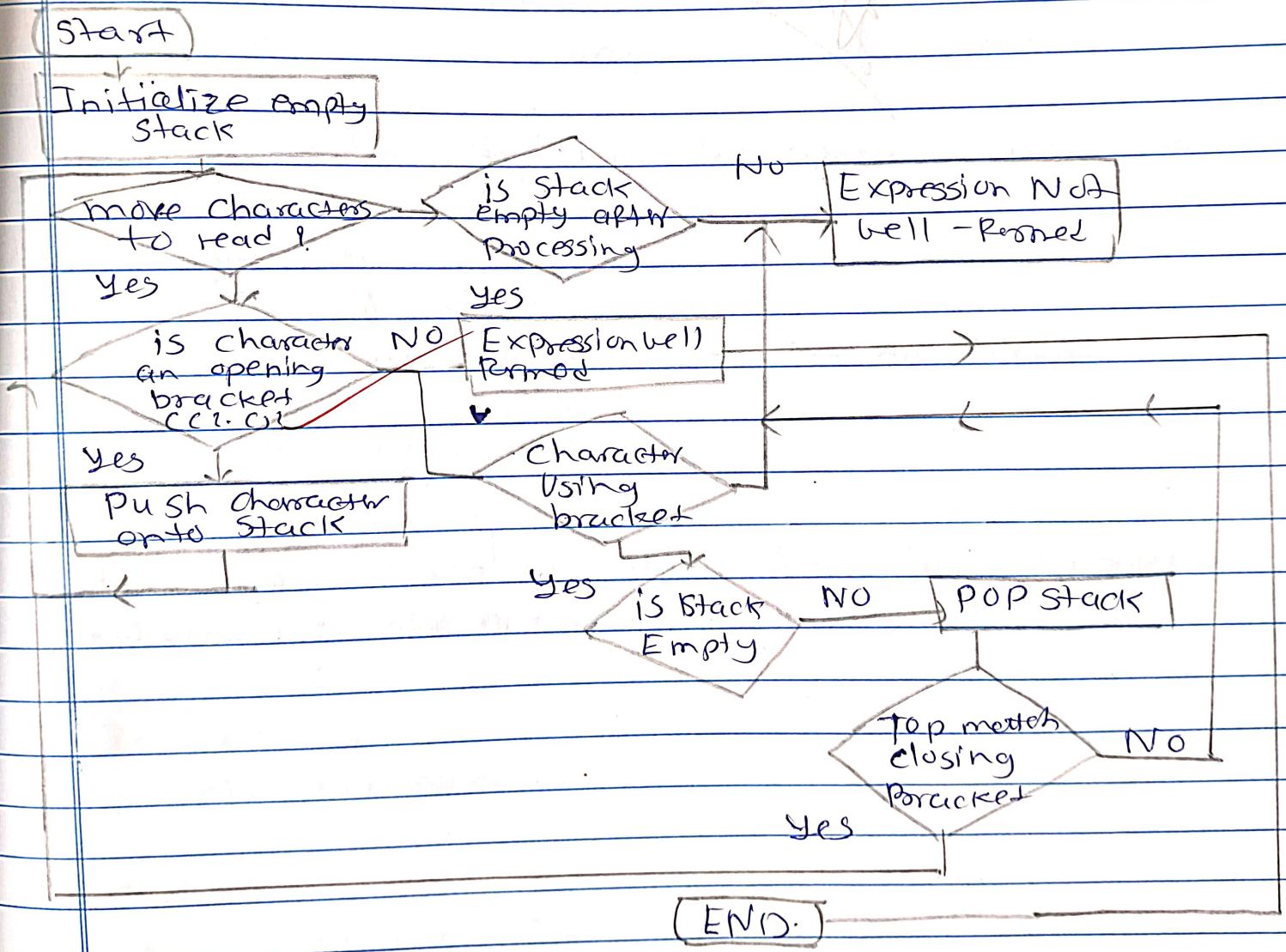
```
std :: string expression;
std :: Cout << "Enter an expression with parenthesis";
std :: Cin >> expression;
If C is well parenthesized (expression))
{
std :: Cout << "The expression is well parathesized
  << std :: endl;
}
else
{
std :: Cout << "The expression is not well Parethesized"
      << std :: endl;
}
return 0;
}
```

Algorithm :-

1) Initialize a stack keep track of the opening brackets.
2) Traverse each character in the input expression.
   i) push it onto the stack.
   ii) check if the stack is empty:
   if it is return false because there is no matching opening bracket.
   iii) otherwise, pop the top element for the stack.

3) After traversing the entire expression
- If the stack is empty, return true because all brackets were matched currently.
- If the stack not empty return False.

* Flowchart :-



```
case '+' : S.push (val1 + val2); break;
case '-' : S push (val1 - val2); break;
Case '*' : S push (val1 * val2) ; break;
ase '/' : S. Push (val1 / val2); break;
}
}
}
return Stop ();
}
int main ()
{
String infix;
Cout << " Result of Evaluation : " << result <<
    endl;
retar 0;
}
```

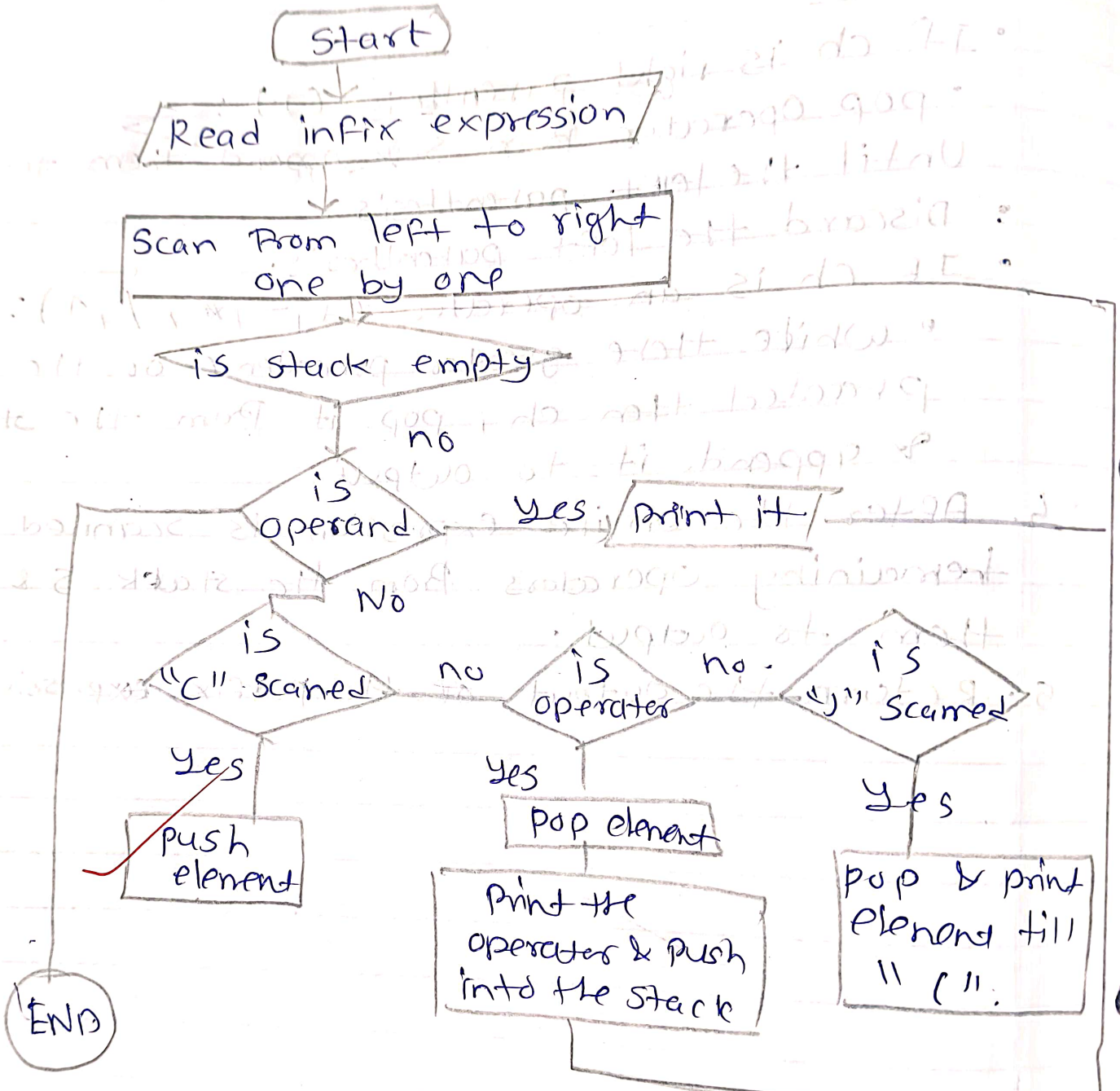
* Algorithm :-

1. Initialize an empty stack S for operaters.
2. Initialize an empty list output to store the postfix expression.
3. For each characters ch in the infix expression (from left to right).
   - If ch is an operand, append it to output
   - If ch is a left parenthesis (c), push it onto the stack S.

12-D-8

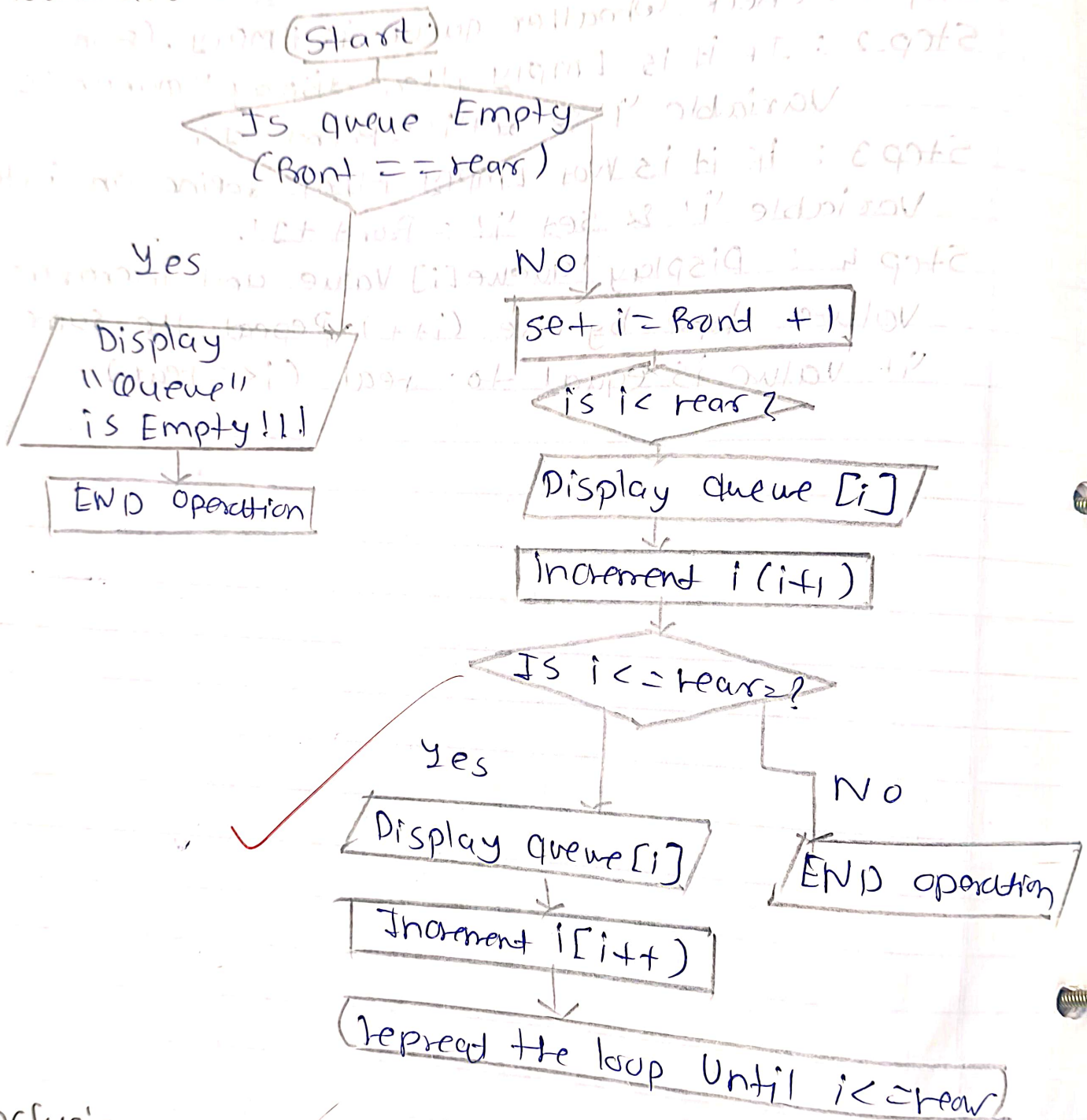
- If ch is right parenthesis ( )):
  - pop operator from S & append them to output until the left parenthesis.
  - Discard the left parenthesis.
- If ch is an operator (+, −, *, /, ↑):
  - while there greater precedence or the same preceded then ch, pop it from the stack & append it to output.
4. After the entire expression is scanned, popall remaining operators from the stack S & append them to output.
5. Return the output of the postfix expression.

Flowchart :-



Conclusion :- By this way we can perform expression Conversion as infix to postfix & its evalution Using stack

Flowchart :-

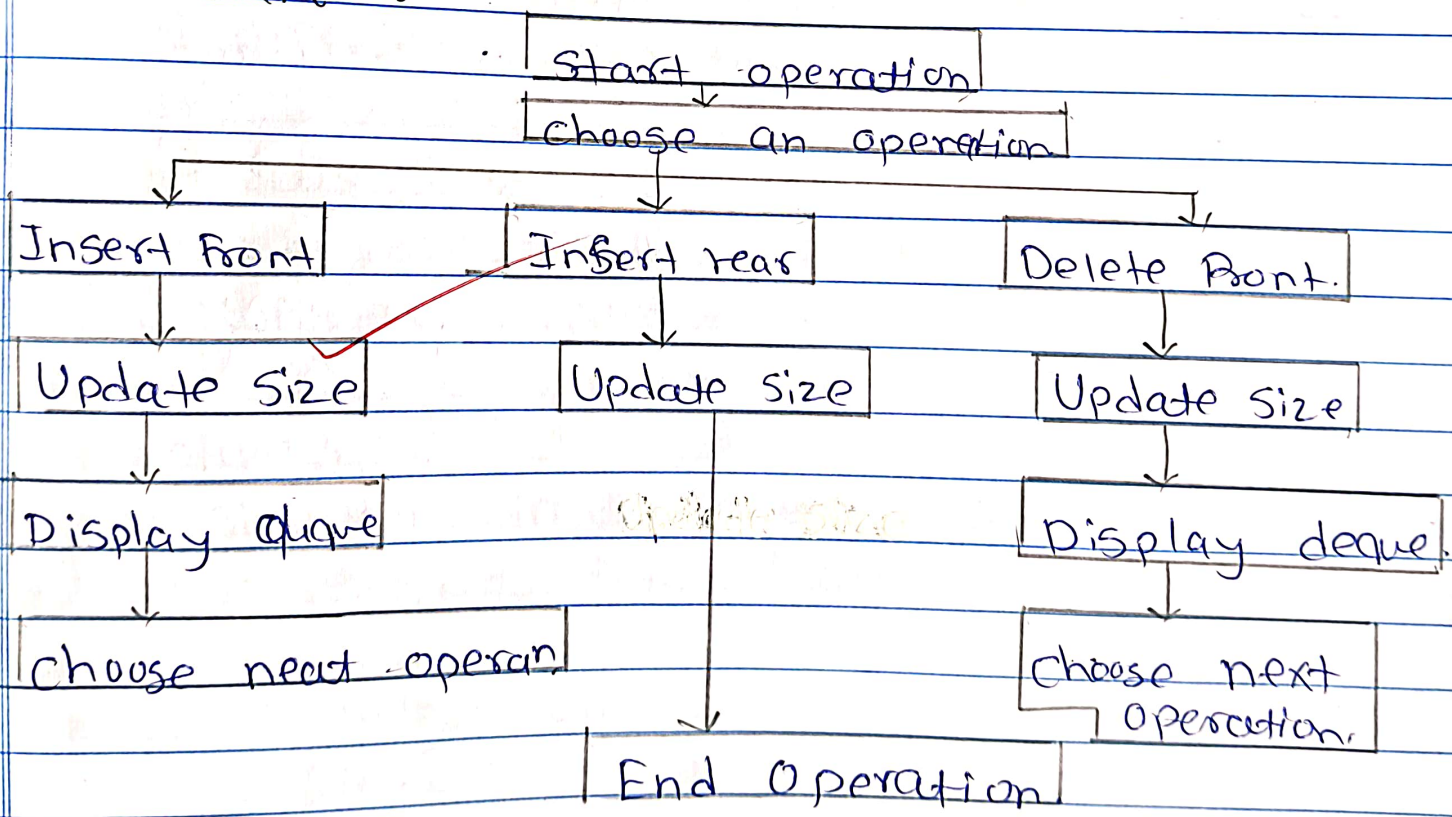


Conclusion : By this way we can perform different operations on queue.

Algorithm :-

1. Initialize deque.
2. Check if deque is full.
3. Check of deque if empty
4. Insert at Front
5. Insert at rear.
6. Delete From Front.
7. Delete From Rear
8. get Front element.
9. get Rear element.

Flowchart :-



2. Else :
   - IF front == -1 (initial state) ; set front = 0
   - Increment rear = (rear + 1) % MAX
   - Insert the new element at queue[rear]
3. End.

Algorithm For Dequeue :-

1. Check Underflow
   - If front == -1 ; the queue is empty . print "queue is Empty" & terminate.

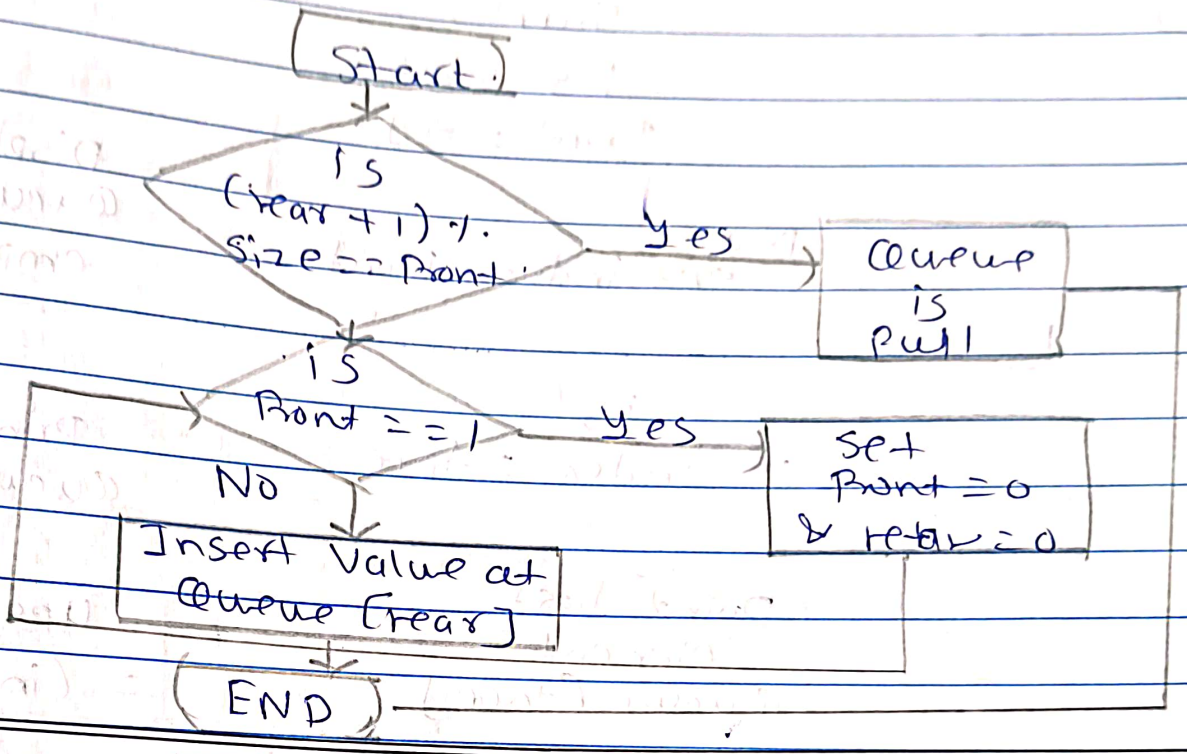
2. Else :
   - Retrieve the element at queue[front]
   - If front = rear, set front = rear = -1
   - Else ; set front = (front * 1) % MAX
   - Print the requested element

3. END.

Flowchart.

ISE-7

(1)



(2)