Algorithm:

Step 1:- Staxt
Step 2: define class complex
-data membex float x(real pairx), float y
-constructox : default.

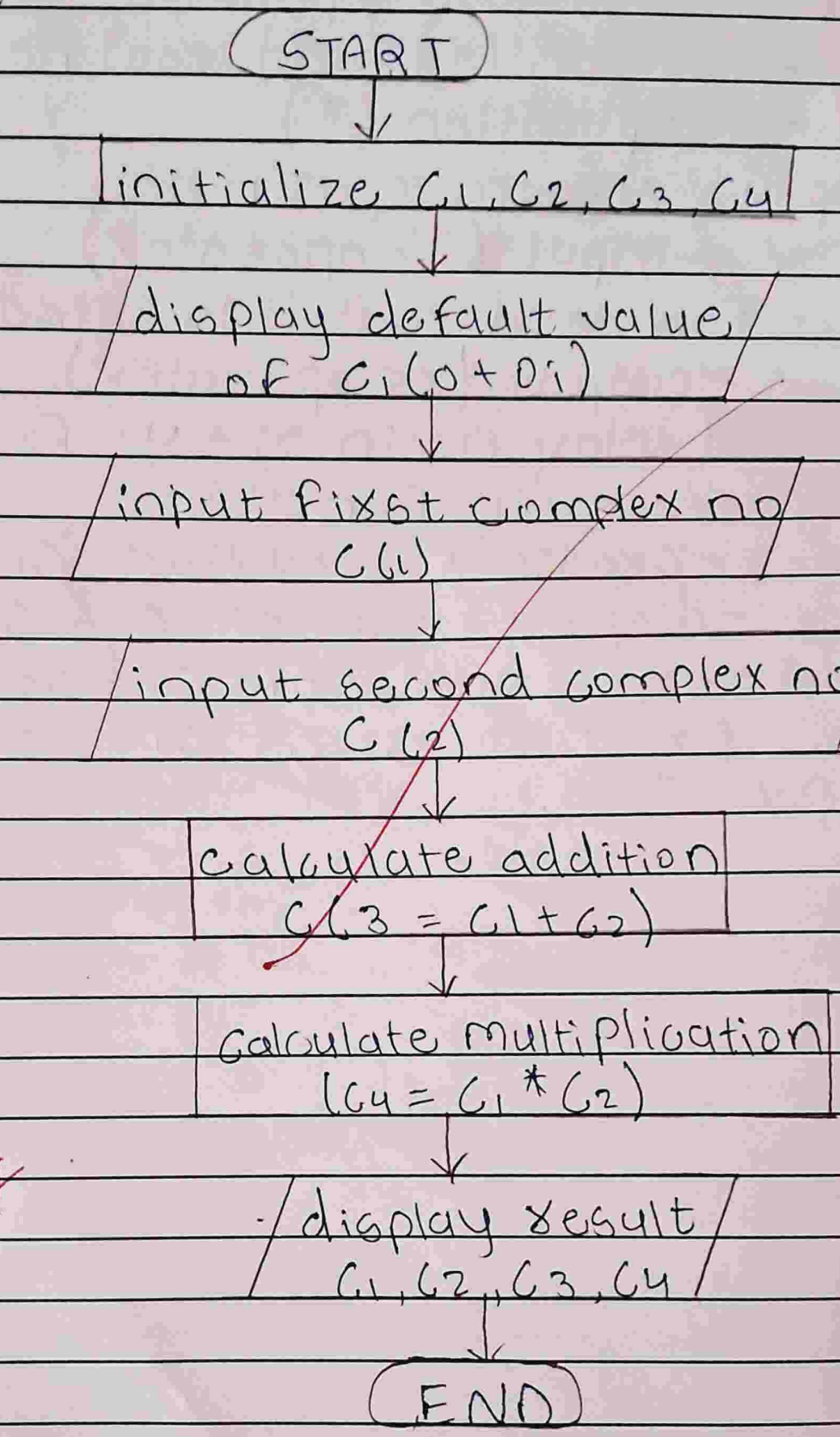
Step 3 : overload operator
- Addition (+) ; add real no & imaginary
- multiplication (*)

step-4 : stream operator
- input (>> operator)
for read real & imaginary part
- output (<< operator)
display no in x + y: from

Step-5 : main Function
- create instruction C1, C2, C3 & C4
- display value of C1 & C2
- input value of C1 & C2
print C1, C2, C3, C4

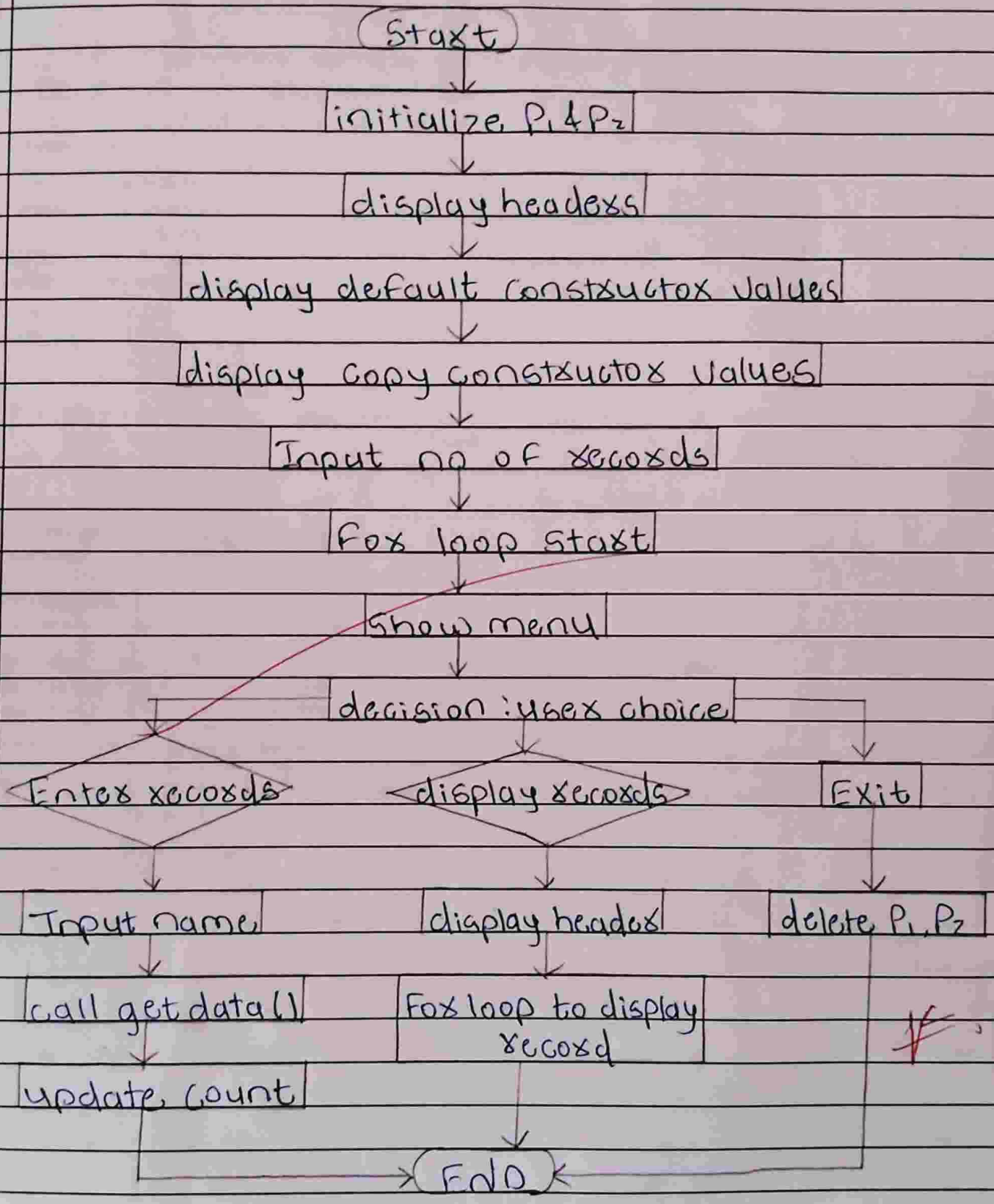
Step-6 : Stop.

Flowchart:-


START
initialize C1, C2, C3, C4
display default value of C1(0+0i)
input first complex no C(1)
input second complex no C(2)
calculate addition C(3 = C1+C2)
calculate multiplication (C4 = C1 * C2)
display result C1, C2, C3, C4
END

Algorithm

1) Start
2) define classes
- Publication
- Attributes: title (string), price (float)
- constructor: initialize title & price
3) :books (inherits from Publication)
- Attributes Pages (int)
• initialize title, price & pages
• get b() to get details is bet 500 - 1500
-if valid display back details
4) Tape
-min (float) initialize title price & min
get t() to get details
if min is not bet 30-90 & price 500 -1000.
-if valid display details.
5) Main function
-Ask if they want buy book (1) or tape (2)
-if 1, call g.get() for bank
if 2, call g.get() for tape
catch exception
6) END.


Start
initialize P1 & P2
display headers
display default constructor values
display copy constructor values
Input no of records
For loop start
Show menu
decision : user choice
Enter records
display records
Exit
Input name
call getdata()
update count
display header
For loop to display record
delete P1, P2
End

GURU GOBIND SINGH FOUNDATION
POLYTECHNIC [ ] / DEGREE [ ]

Roll No. :
Page :
Date :

Algorithm:

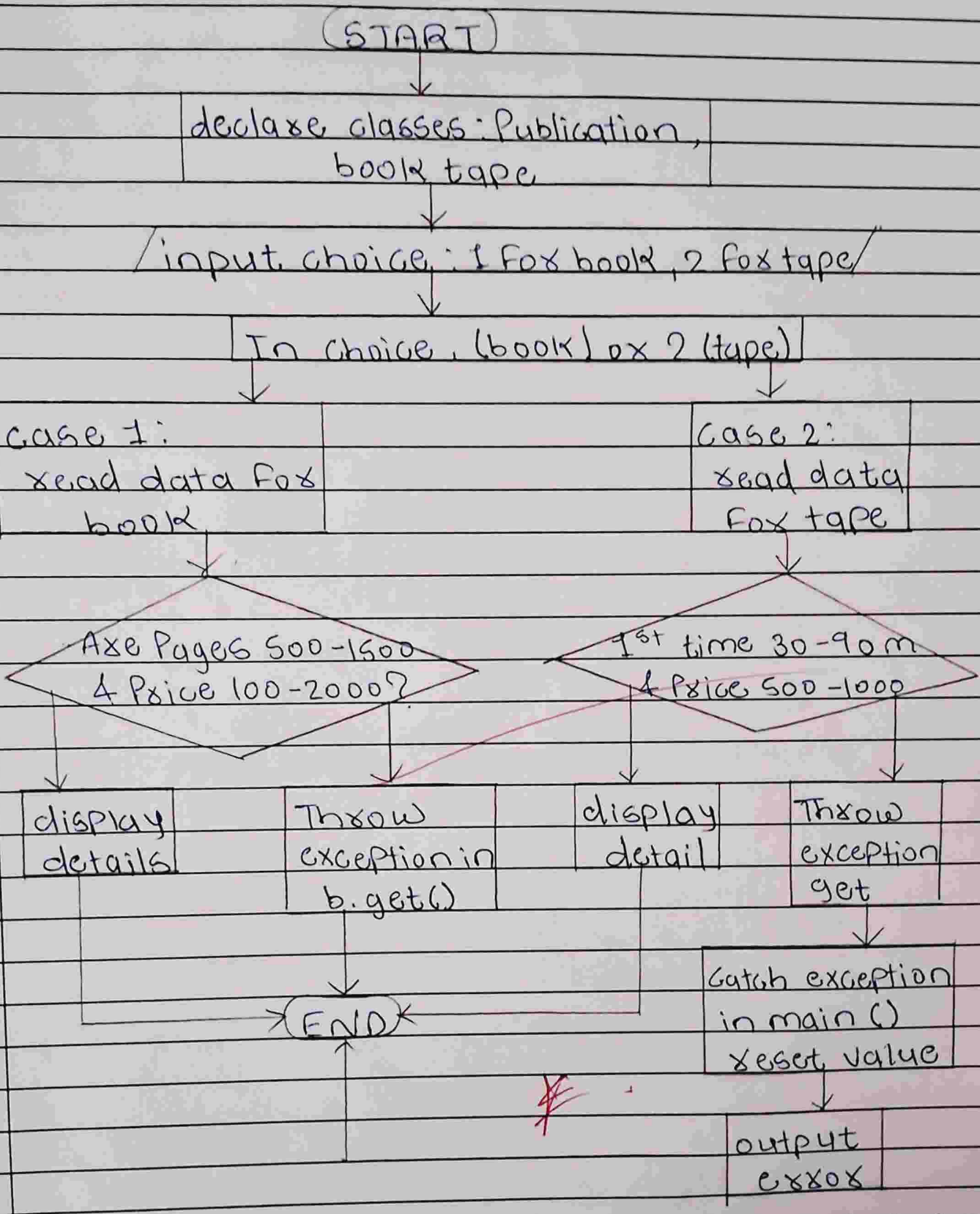
Step 1: Start
Step 2: Create P_1 & P_2 object of person class
Step 3: display header & details for P_1 & P_2
Step 4: main menu operation.
- input no of records n
- create an array: P_3 of person objects.

Step-5: menu loop
- display menu option
- input user choice ch
if ch = 1
- Input name
- call $P_3[n]$ get data (name)

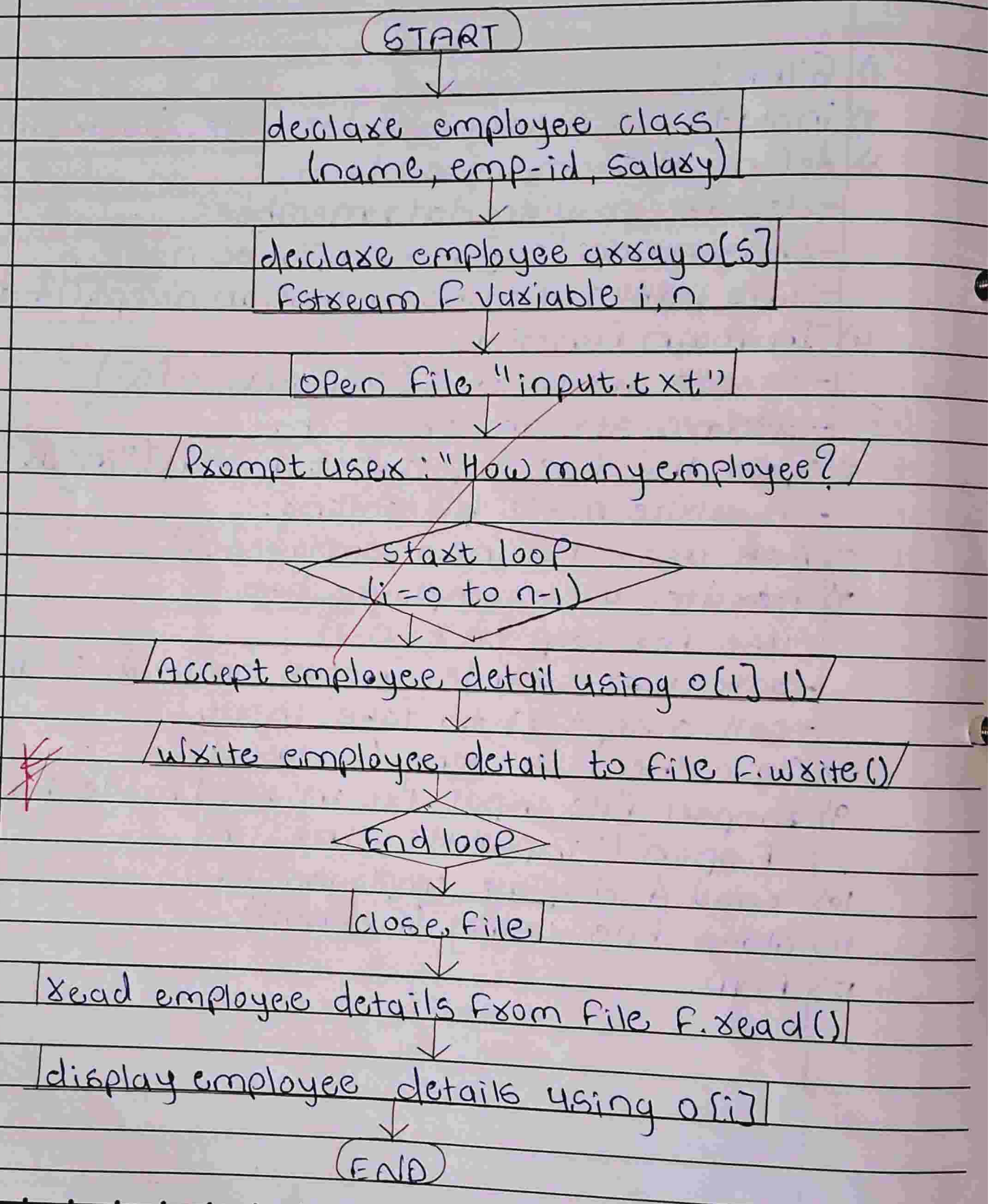
Step 6: if ch == 2
- display headers
- For each records in B_3, call $P_3[i]$
if ch == 3
exit the loop

Step 7: clean up
- delete P_1, P_2, P_3

Step 8: END

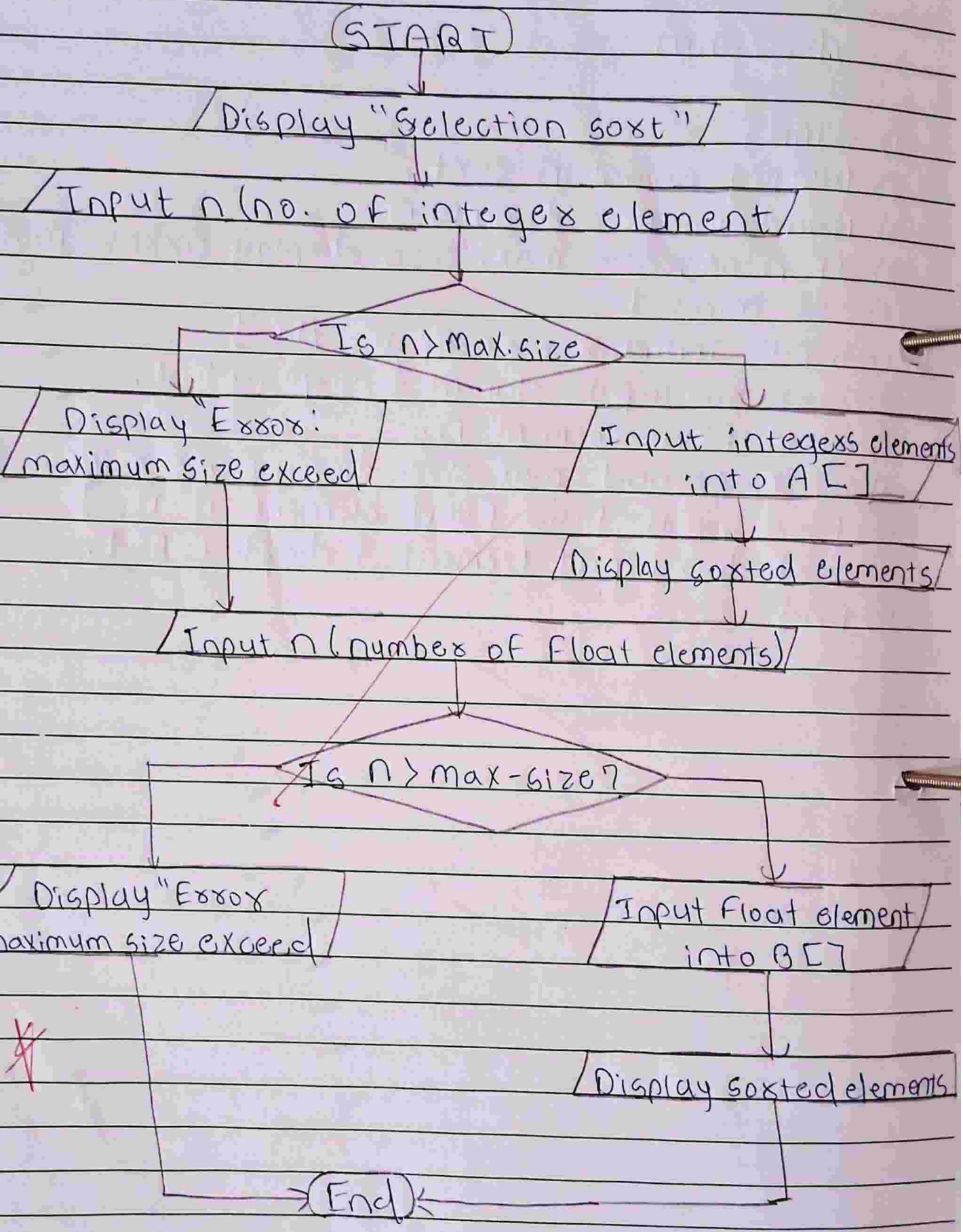

START
declare classes: Publication, book, tape
/input choice: 1 for book, 2 for tape/
In choice, (book) or 2 (tape)
case 1: read data for book
case 2: read data for tape
Are Pages 500-1500 & Price 100-2000?
1st time 30-90 m & Price 500-1000
display details
Throw exception in b.get()
display detail
Throw exception get
Catch exception in main () reset value
output error
END

1) START
2) include necessary header file (iostream & fstream)
3) define the employee class:
   - declare private data members
   - char name [20] for employee name.
   - Two public member function accept() & display()
4) In main function
   - create array 0-5 (employee e[5])
   - declare stream object f.
5) Open the file input.txt f.open ("input.txt")
   - in write mode by calling.
6) Ask user for no of employee (n)
7) iterate over employee list.
   - use for loop (0 to n-1)
   - ask user to enter (name, emp-id, salary)
   - call accept () to take input.
8) Close the file using (f.close())
9) reopen file input.txt in read mode by calling
   f.open ("input.txt, ios::in)
10) read & display employee info -
11) Close file using (f.close())
12) END.


START
declare employee class (name, emp-id, salary)
declare employee array o[5] fstream f variable i, n
open file "input.txt"
Prompt user : "How many employee?
Start loop (i=0 to n-1)
Accept employee detail using o[i] ()
write employee detail to file f.write()
End loop
close file
read employee details from file f.read()
display employee details using o[i]
END

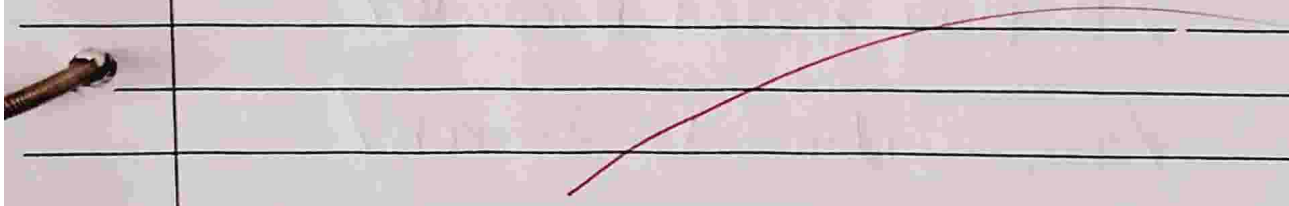
Algorithm

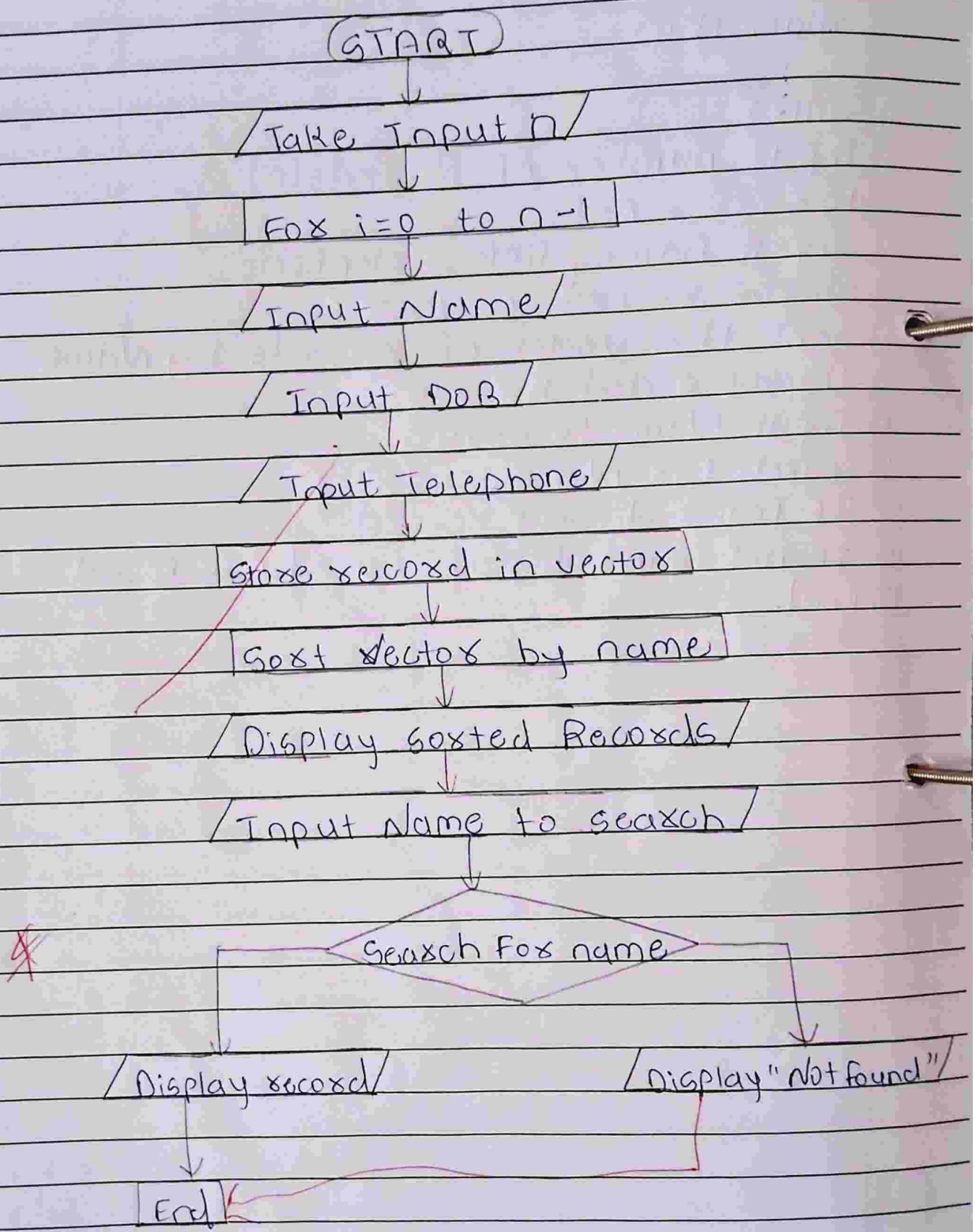
1) START
2) Display selection sort.
3) Read integer elements n.
4) IF n exceeds max. size display error message
5) Set min = 1
6) input integer element in A[]
7) Sort the input element & display list.
8) Read Float element n.
9) IF n exceed max-size display error message
10) Else input take Float element in B[]
11) Sort the input element & display list.
12) End.


START
Display "Selection sort"
Input n (no. of integer element)
Is n > max. size
Display "Error: maximum size exceed
Input integers elements into A[]
Display sorted elements
Input n (number of float elements)
Is n > max-size?
Display "Error maximum size exceed
Input Float element into B[]
Display sorted elements
End

Algorithm

1) START
2) Input number of Records(n)
3) For i=0 , n-1
4) Input name, DOB, Telephone
5) Stores record in vector
6) Sort the vector of records by Name
7) Display sorted Records
8) Input Name to search
9) Search for Name in vector
10) If found, display record
11) If not found display "Record not found
12) End.

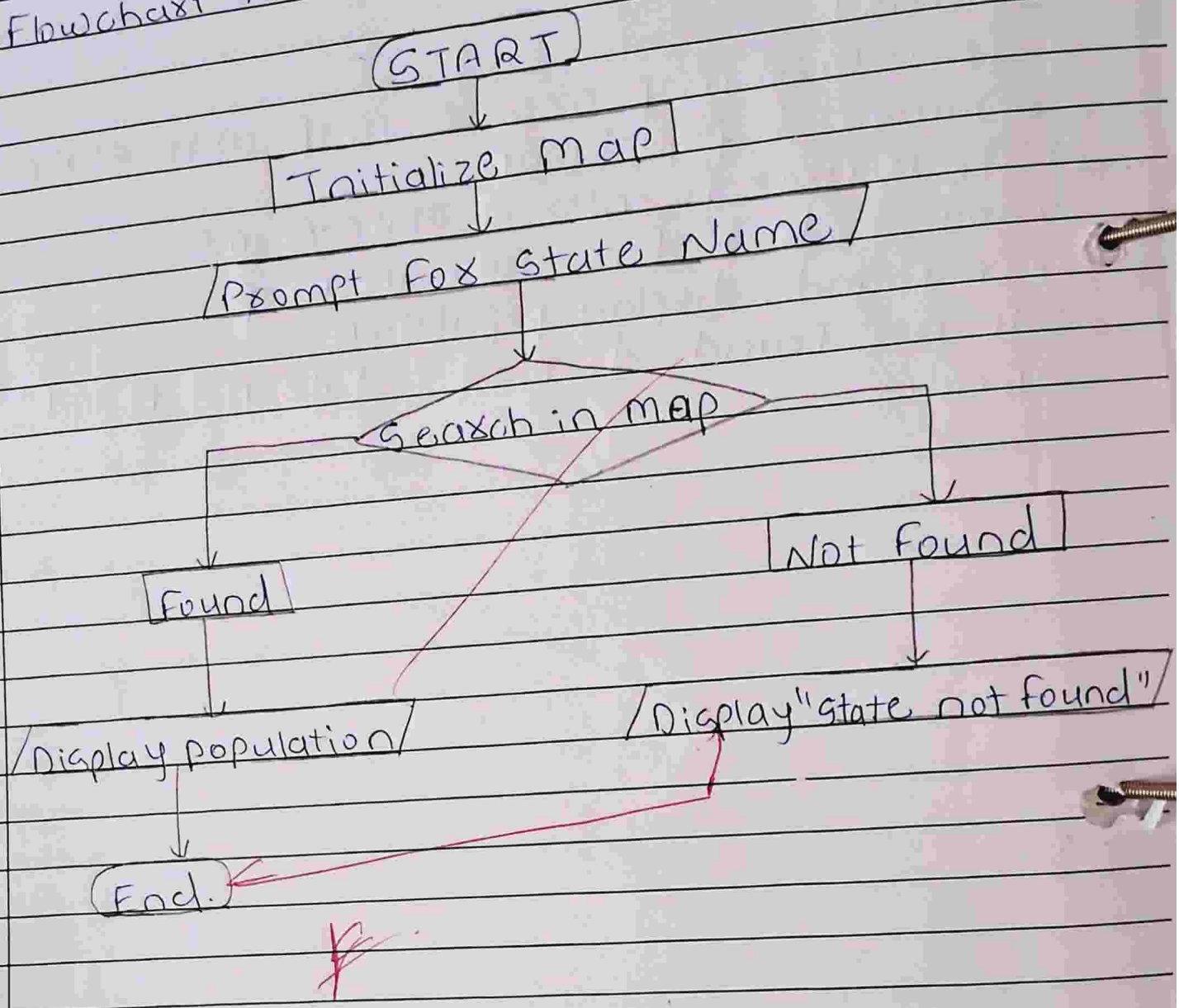



START
Take Input n
For i=0 to n-1
Input Name
Input DOB
Input Telephone
Store record in vector
Sort Vector by name
Display Sorted Records
Input Name to search
Search For name
Display record
Display "Not found"
End

Algorithm

1) START
2) Initialize map create map with state names & population
3) Input user to enter a state name
4) Search look for state in map
5) If found, display population
6) If not found, display "State not found".
7) End.

Flowchart :


START
Initialize map
Prompt for state Name
Search in map
Found
Not Found
Display population
Display "state not found"
End.