NAME: AYUSH PRASHANT KALASKAR.

CLASS: S.E. COMPUTER SCIENCE DIVISION: S.E. COMP-I

ROLL NO. : 69

SUBJECT : DATA STRUCTURE LAB (DSL)

ASSIGNMENT : B-11

* ALGORITHM:- For Linear and Sentinel Search

(1.)

1. Intialize an empty list students_attend
2. Prompt the user to enter the total number of students who attended the training program.(n)
3. Create a for loop to read and store the roll number of students who attended:
   a. Intialize an empty list students_attend.
   b. For i in range(n):
      i. Prompt the user to input a roll number.
      ii. Convert the input to an integer and append it to the list students_attend
4. Prompt the user to enter the roll number to search (search_element)

2. Algorithm for Function linear_search()

→

1. Input: students_attend (a list of student roll numbers)
   search_element (an integer)

i.e. linear_search(students_attend, search_element)

2. For each roll_number in students_attend:
   a. If roll_number is equal to search_element, return 1 (indicating Found.)
3. If the loop completes without finding the search_element, return -1 (indicating not found.)

(3.) Algorithm for function sentinel_search():

sentinel_search(students_attend, search_element, n)

→ 1. Input: students_attend (a list of roll number of stude
, search_element (an integer),
n (the total number of students.)

2. Intialize last as the last element of students_atten
(students_attend[n-1])

3. Set students_attend[n-1] to search_element (placing the search_element at the end of the list as a sentinel.)

4. Intialize i=0.

5. While students_attend[i] is not equal to search_element:
   a. Increment i by 1.

6. Restore the original last element by setting students_attend[n-1] back to last.

7. If (i < n-1) or (search_element is equal to student_attend[n-1]), return 1 (indicating found)

8. Otherwise, return -1 (indicating not found.)

(4.) Algorithm for Displaying Menu:-

→ 1. Display a menu with two search options: Linear Search and Sentinel Search.

2. Prompt the User to Enter your choice (ch).

3. If ch is 1 (for Linear search):
   a. Call the linear_search Function with students_attend and search_element.
   b. If the result is -1, print The student didn't attendend the program.
   c. Otherwise, print " The Ent student has attended the training program.

4. If ch is 2 (For Sentinel search):

→ a. Call the sentinel search function with students_attend, search_element, and n.

b. If the result is -1, print The student didn't attended the training program.

c. Otherwise, print The student attended the training program.

5. If the user enters an invalid choice, print Wrong Choice.

* ALGORITHM :- For Binary and Fibonacci Search()

(1.)

1. Intialize an empty list (students_attend.
2. Prompt the user to enter the total number of students who attended the program (n):
3. Create a For loop to read and store the roll number of students who attended:
   a. Intialize an empty list students_attend.
   b. For i in range(n):
      i. Prompt the User to input a roll number.
      ii. Convert the input to an integer and append it to the students_attend List.
4. Prompt the user to ~~search~~ Enter the Roll Number to search (search_element)
5. Sort the students_attend list in ascending order
6. Intialize start to zero.

(2.) Algorithm for binary_search_recursive()

binary_search_recurssive (students_attend, search_element, start, n)

→ 1. Input : students_attend (A sorted List of student roll numbers), search_element (an integer),

, start (start index), n (end index)

2. If start is greater than n, return -1 (indicating not found.)
3. Calculate mid as (start + n) // 2.
5. If search_element is ~~equal to~~ less than students_attend[mid], recursively call binary_search_recursive with updated parameters (start, mid - 1)
6. If search_element is equal to students_attend[mid], return mid (indicating found.)
6. If search_element is greater than students_attend[mid], recursively call binary_search_recursive with updated parameters (mid + 1, n)

(3.) Algorithm for function Fibonacci_search(*)

fibonacci_search(students_attend, search_element, n)

→ 1. Input: students_attend (a sorted list of student roll numbers), search_element (an integer), n (the total number of students)
2. Intialize fibMMm2 as 0, fibMMm1 as 1 and fibM as fibMMm2 + fibMMm1
3. Find the largest fibonacci number that is less than or equal to n and update fibMMm2, fibMMm1 and fibM accordingly.
4. Intialize offset as -1.
5. While fibM is greater than 1:
   a. Calculate i as the minimum of (offset + fibMMm2) and (n-1)
   b. If students_attend[i] is less than search_element, update fibM, fibMMm1, fibMMm2, and offset accordingly.
   c. If students_attend[i], is greater than search_element,

update fibM, fibMMm1, fibMMm2, and offset accordingly.

d. If students_attend[i] is equal to search_element, return offset + 1 (indicating found.)

e. If ~~the elem~~ fibMMm1 is not 0 and students_attend[offset+1] is equal to search_element, return offset + 1 (indicating found.)

7. If the element is not found, return -1 (indicating not found.)

(4.) Algorithm to Display Choice / Menu.

1. Display a menu with two search Options : Binary Search and Fibonacci search.
2. Prompt the user to enter the choice.
3. If ch is 1 (for Binary Search):
   a. Call the Binary_search_recursive function with students_attend, search_element and n.
   b. If the result is -1, print the Entered roll number didn't attended the program.
   c. Otherwise print The Entered Roll Number has attended the program.
4. If ch is 2 (for Fibonacci Search):
   a. Call the fibonacci_search function with students_attend, search_element and n.
   b. If the result is -1, print the Entered Roll Number didn't attended the program.
   c. Otherwise print The Entered Roll Number has attended the program.
5. If the user enters an invalid choice, print Sorry! Wrong Choice.

QUESTIONS:-

Q.1] Compare all Searching Algorithms with its Time Complexity (Write answer in Tabular Format)

ANS. TABLE:-

SR. NO.:-	ALGORITHM:-	TIME COMPLEXITY	SPACE COMPLEXITY	NOTE
1.	Linear Search	O(n)	O(1)	Simplest search Algorithm. Scans the list sequentially.
2.	Sentinel Search	O(N)	O(1)	Adds a sentinel value at the end of the array which is equal to target value we are looking for
3.	Binary Search (Recursive)	O(log N)	O(log N)	Requires a sorted list, divides the list into half. Recursive version of binary search
4.	Fibonacci search	O(log n)	O(1)	Divides the list into Fibonaccis numbers ratio.

Page No.
Date

DSL ASSIGNMENT : B-11

(1.) Pseudocode for the creating and appending ~~search element~~ students to the list and searching for search_element.

→ 1. Create a list students_attend = []
2. Input (Integer) The total students who attended the training program.
3. Print the Roll Number of students who attended the training program.
4. for i in range(n):
(start) students_attend.append(int(input()))
5. Input (Integer) search_element = Roll number of student to search in the list

(2.) Pseudocode for the function
def linear_search(students_attend, search_element):

→ 1. for i in students_attend
(start) if i == search_element then
return 1
else
return -1

NOTE:- students_attend, search_element are input to the function linear_search():

(3.) Pseudocode for the function.
def sentinel_search(students_attend, search_element, n)

→ 1. Input: students_attend (list of student Roll Number)
search_element (An Integer) and
n (The total number of students)
2. Store: last = students_attend[n-1]
3. Store students_attend[n-1] = search_element
Intialize i = 0

```
4. while (students_attend[i] != search_element):
        i += 1
5. students_attend[n-1] = last
   if (i<n-1) or (search_element == students_attend[n-1]) then
       return 1
   else
       return -1
```

(4) Pseudocode for the displaying Option list / Menu.

→
```
1. print Linear Search
   print Sentinel Search
2. input ch = Enter your choice.
3. if ch == 1 then
   result = linear_search(students_attend, search_element)
   if result == -1 then
       print The Entered Roll no. did not attend the
                   training program.
   else
       print The Entered Roll no. has attended the
                   training program.
4. if ch == 2 then
   result = sentinel_search(students_attend, search_element,n
   if result == -1 then
       print The Entered Roll no. did not attend the
                   training program
   else
       print The Entered Roll no. has attended the
                   training program.
5. else
   print Sorry! Wrong Choice.
```

DSL - ASSIGNMENT : B-11

(1.) Pseudocode for creating and appending roll no. of students to the list and searching the search_element in it.

→ 1. Create a List: students_attend = []
2. Input 'n' (Interger) i.e. Total Number of students who attendend the program
3. print The Roll Number of students who attended the program.
4. for i in range(n):
   students_attend.append(int(input()))
5. search_element = int(input("Enter the roll no. you want to search"))
6. students_attend.sort()
7. Intialize start = 0.

(2.) Pseudocode for the function
binary_search_recursive(students_attend, search_element, start, n):

→ 1. Input: students_attend (list of roll no. of students), search_element (An Integer), start (start index) and n (Last Index.)
2. If start > n then
   return -1
   mid = (start + n) // 2
   If search_element == students_attend[mid] then
   return mid
   if search_element < students_attend[mid] then
   return.
   binary_search_recursive(students_attend, search_element, start, mid-1)
   else
   return
   binary_search_recursive(students_attend, search_element, mid+1, n)

(3.) Pseudocode for the function:

fibonacci_search(students_attend, search_element, n):

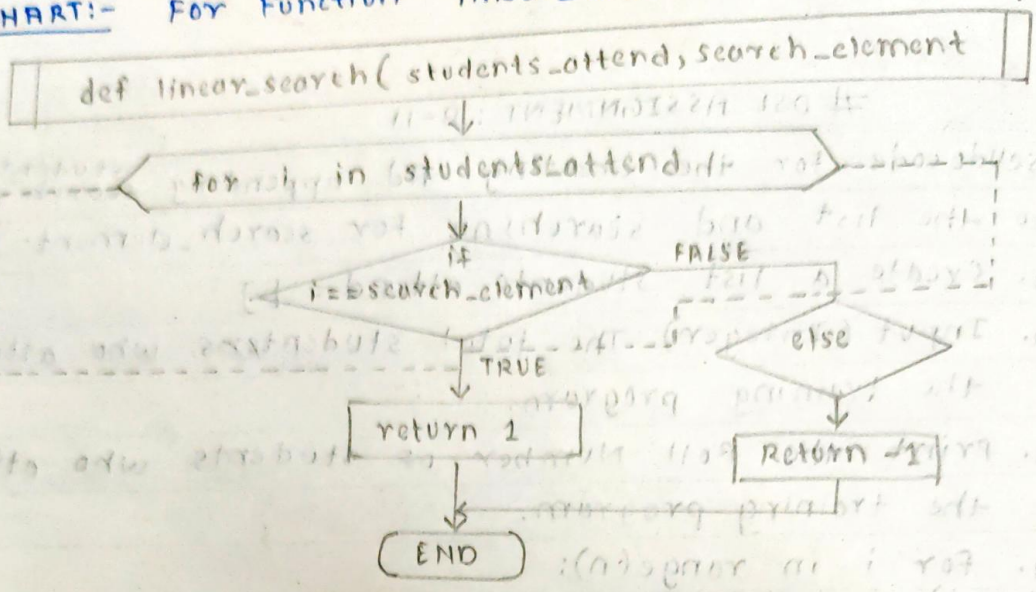
1. Intialize : fibMMm2 = 0
   fibMMm1 = 1
2. store : fibM = fibMMm2 + fibMMm1
3. while (fibM < n):
   fibMMm2 = fibMMm1
   fibMMm1 = fibM
   fibM = fibMMm2 + fibMMm1.
4. Intialize : offset = -1
5. while (fibM > 1):
   i = min (offset + fibMMm2, n-1)
6. if (students_attend [i] < search_element) then
   fibM = fibMMm1
   fibMMm1 = fibMMm2
   fibMMm2 = fibM - fibMMm1
   offset = i
   elif (students_attend [i] > search_element)
   fibM = fibMMm2
   fibMMm1 = fibMMm1 - fibMMm2
   fibMMm2 = fibM - fibMMm1
   else
   return i
7. if (fibMMm1 and students_attend[offset+=1] == search_element) then
   return offset + 1
   else
   return -1

(4.) Pseudocode for Displaying Menu or Option List.

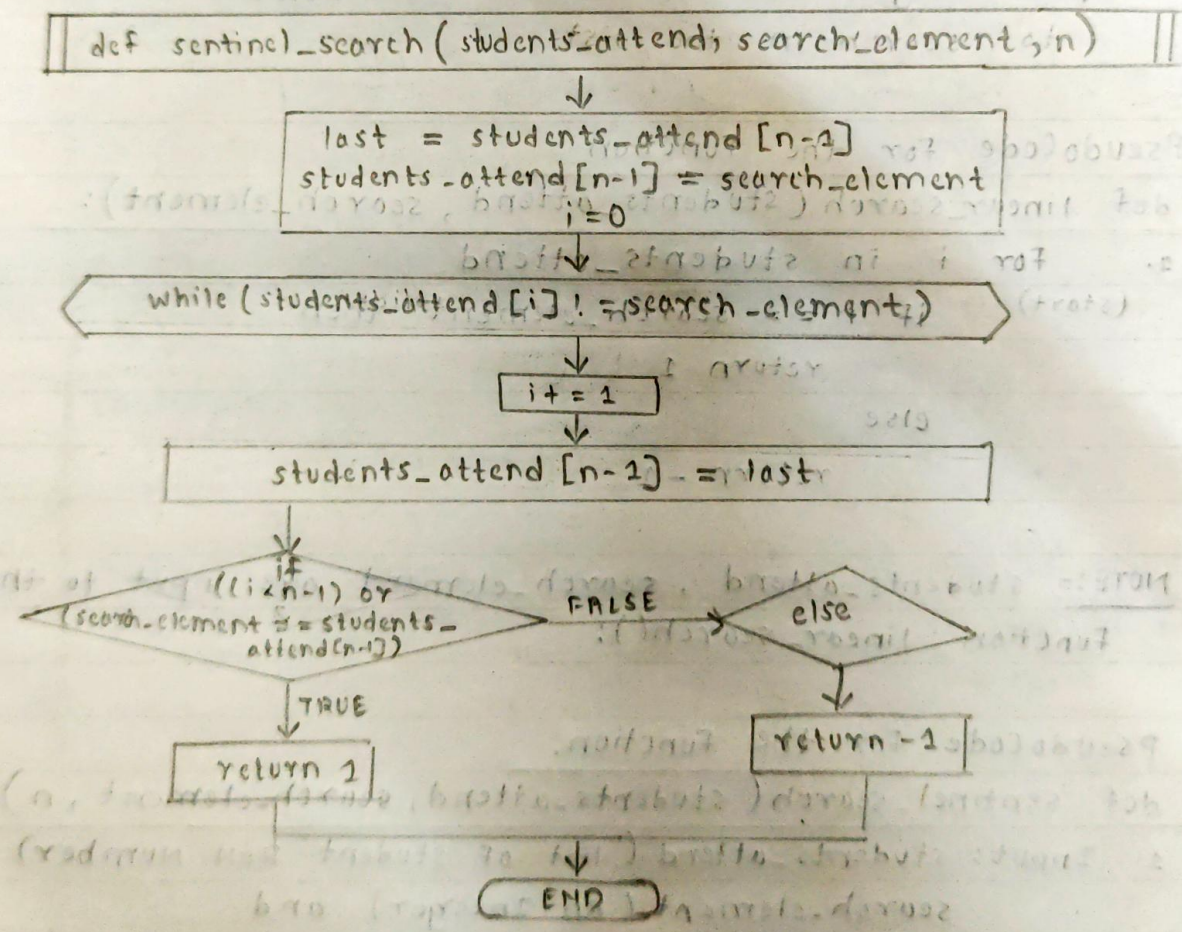
⟹

```
1. print  1.) Binary Search
   print  2.) Fibonacci Search.
2. ch = int(input("Enter your choice : "))
3. if ch == 1 then
     result = binary_search_recursive(students_attend,
                search_element, start, n)
     if result == -1 then
       print The Entered Roll no. did not attend the
             training program.
     else
       print The Entered Roll no. has attended the
             training program.
4. elif ch == 2 then
     result = fibonacci_search(students_attend, search_element,
              n)
     if result == -1 then
       print The Entered Roll no. did not attend the
             training Program.
     else
       print The Entered Roll no. has attended the
             training program.
5. else
     print Sorry! Wrong choice.
```

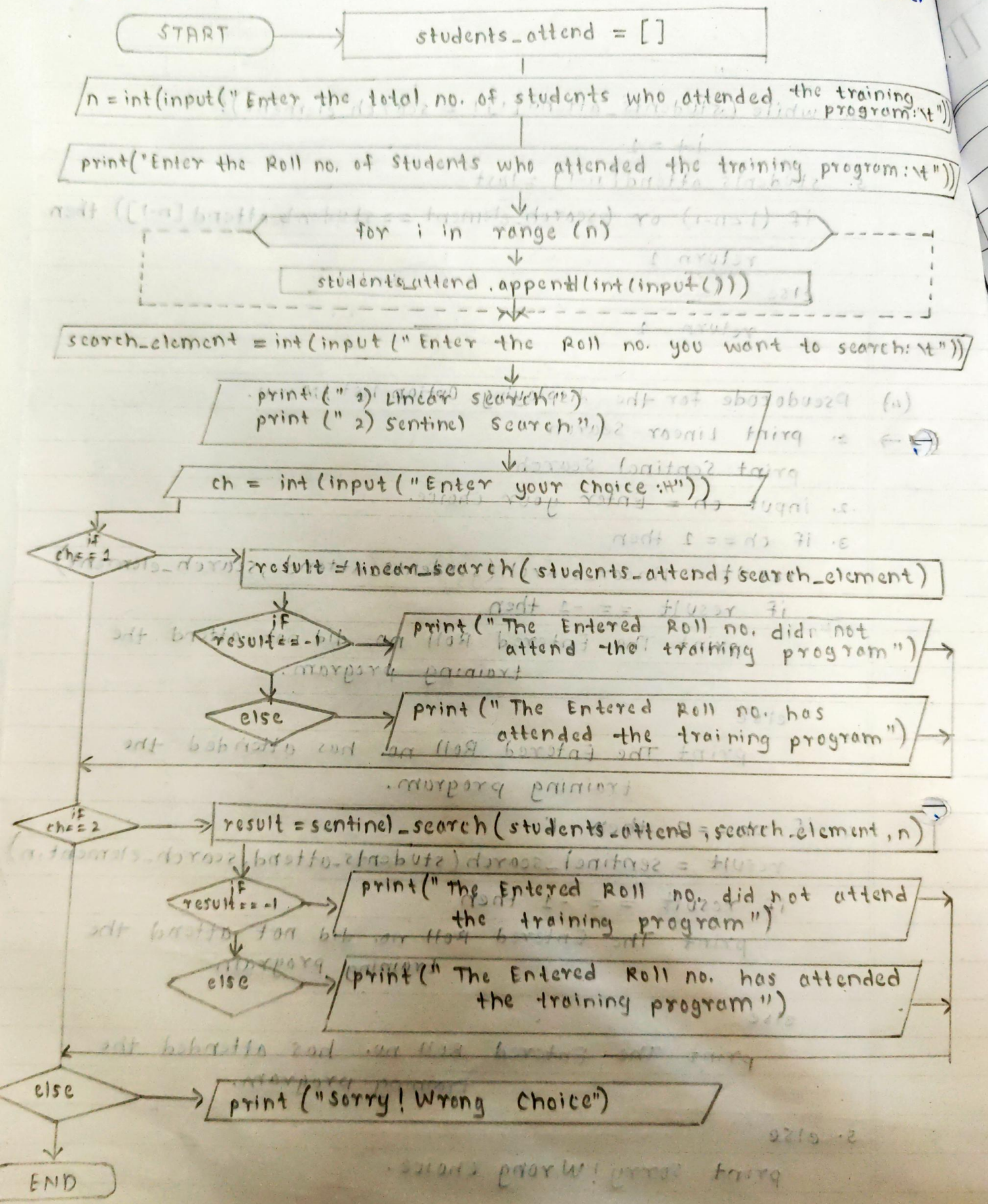
(1.) FLOWCHART:- For Function linear_search():



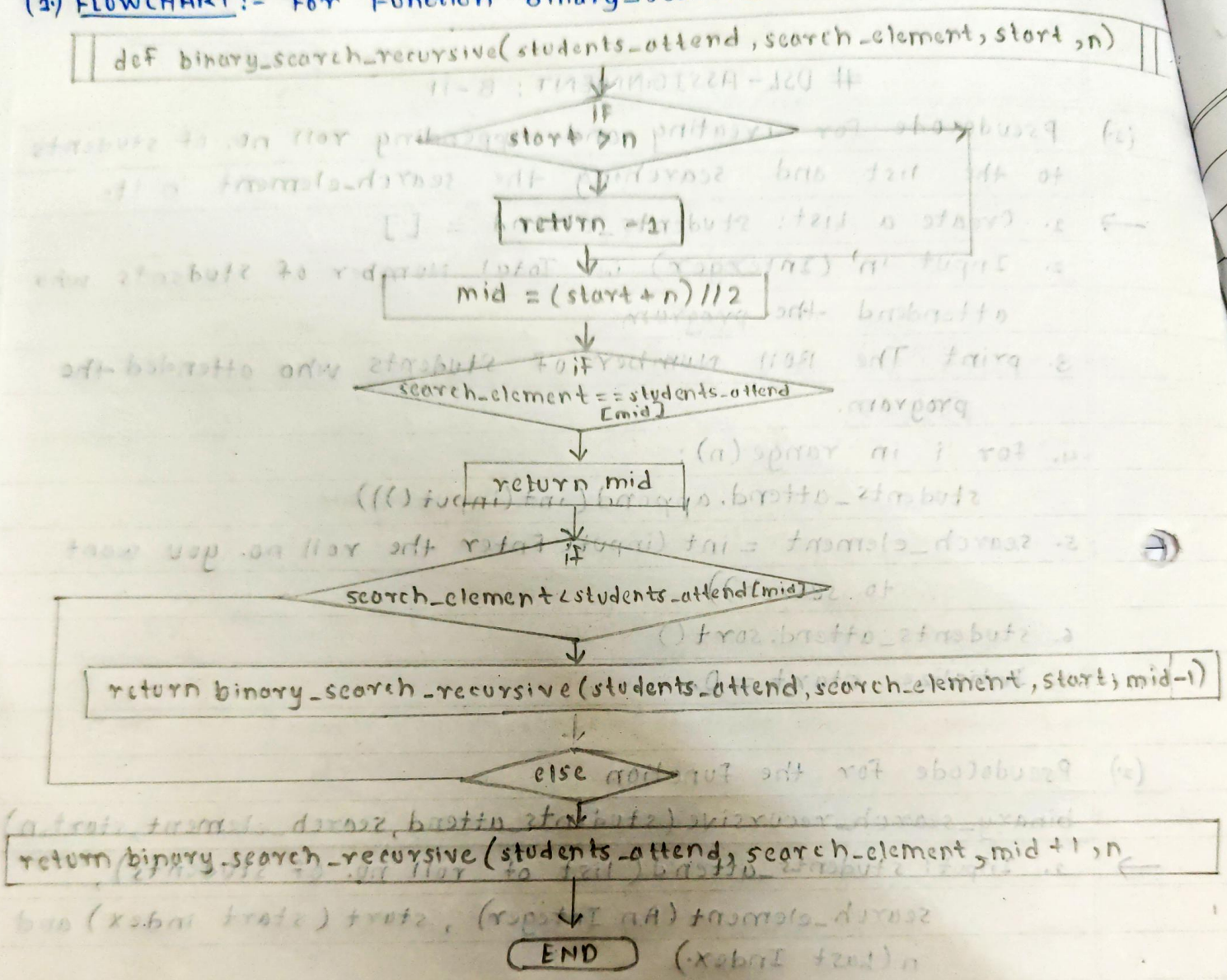
(2.) FLOWCHART:- For Function sentinel_search():



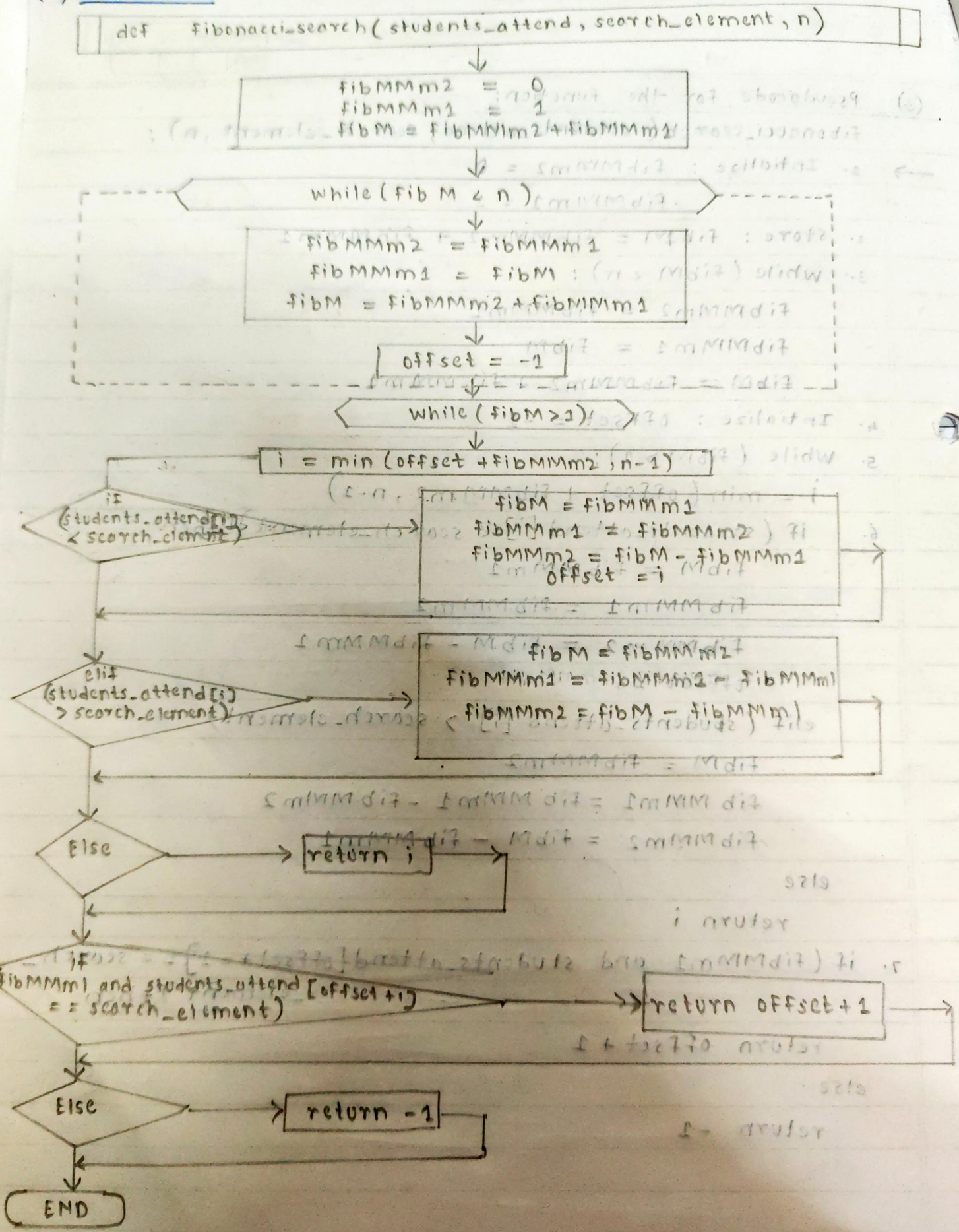
(3.) FLOWCHART FOR THE MAIN PROGRAM:- AND MENU/Option List.



(1) FLOWCHART :- For Function binary_search_recursive():



(2.) FLOWCHART: For the Function: def fibonacci_search():



(3). FLOWCHART:- For the main program code and Menu/Option List