DYP
Dr. D.Y. PATIL EDUCATIONAL COMPLEX
AKURDI, PUNE

* Title :- Implementation of Set theory.

* Objective :-
  - To Understand the concept of arrays, lists.
  - To explore the concept of set theory.

* Problem Statement :- In second year computer engineering class group A students play cricket, group B students play badminton & group C students play football. Write a python program to Compute followings :-

(1) List of students who play both cricket & badminton.
(2) List of students who play either cricket or badminton but not both.
(3) Number of students who play neither cricket nor badminton.
(4) Number of students who play cricket & football but not badminton.

* Outcomes :-
  - After performing this practical, we understand how to implement different functions, arrays & lists.
  - We are able to apply knowledge of mathematics engineering fundamental problem solving skills, algorithmic analysis to solve problem.
  - Thinking ability of us get enhanced.

* Software Requirements :-
1) Python Programming Environment - Python3 (version)
2) 32/64 bit Operating System - Linux OS (which ?)

* Hardware Requirements :-
1) Any CPU with i3 or next version processor.
2) System with 8 GB + RAM
3) And storage of 256 GB +

* Theory :-

1) Lists :- Lists are one of a built-in datatype in python used to store collection of data. list are enclosed in [] & seperated by comma. list are mutable & hence, they can be altered even after their creation.

2) Sets :- Set theory is a branch of mathematical logic that studies sets, which informally are collections of object. Although any type of object can be collected into a set. Set theory is applied most often to objects that are relavent to mathematics.

3) Set Operations :- Sets are collection of distinct object which is unordered, unchangable & unindexed. Sets are also one of a built-in datatype in python used to store collection of data. Sets are enclosed in {} & seperated by comma.

There are many types of set operations as follows:-

(a) Set Union - The union of 2 sets are the set of all the elements of both the sets without duplicates. It is denoted by symbol 'U'

$A = \{2, 3, 4, 5\}$

$B = \{6, 7, 8, 9\}$

$\therefore A \cup B = \{2, 3, 4, 5, 6, 7, 8, 9\}$

(b) Set Intersection - The intersection of 2 sets is the set of all common elements of both the sets. It is denoted by symbol '$\cap$'

$A = \{2, 3, 4, 5\}$

$B = \{3, 5, 6, 7\}$

$\therefore A \cap B = \{3, 5\}$

(c) Set Difference - The difference between 2 sets is the set of all the element in first set that are not present in second one. Difference performed using '-' operator.

$A = \{2, 3, 4, 5\}$

$B = \{3, 5, 6, 7\}$

$\therefore A - B = \{2, 4\}$

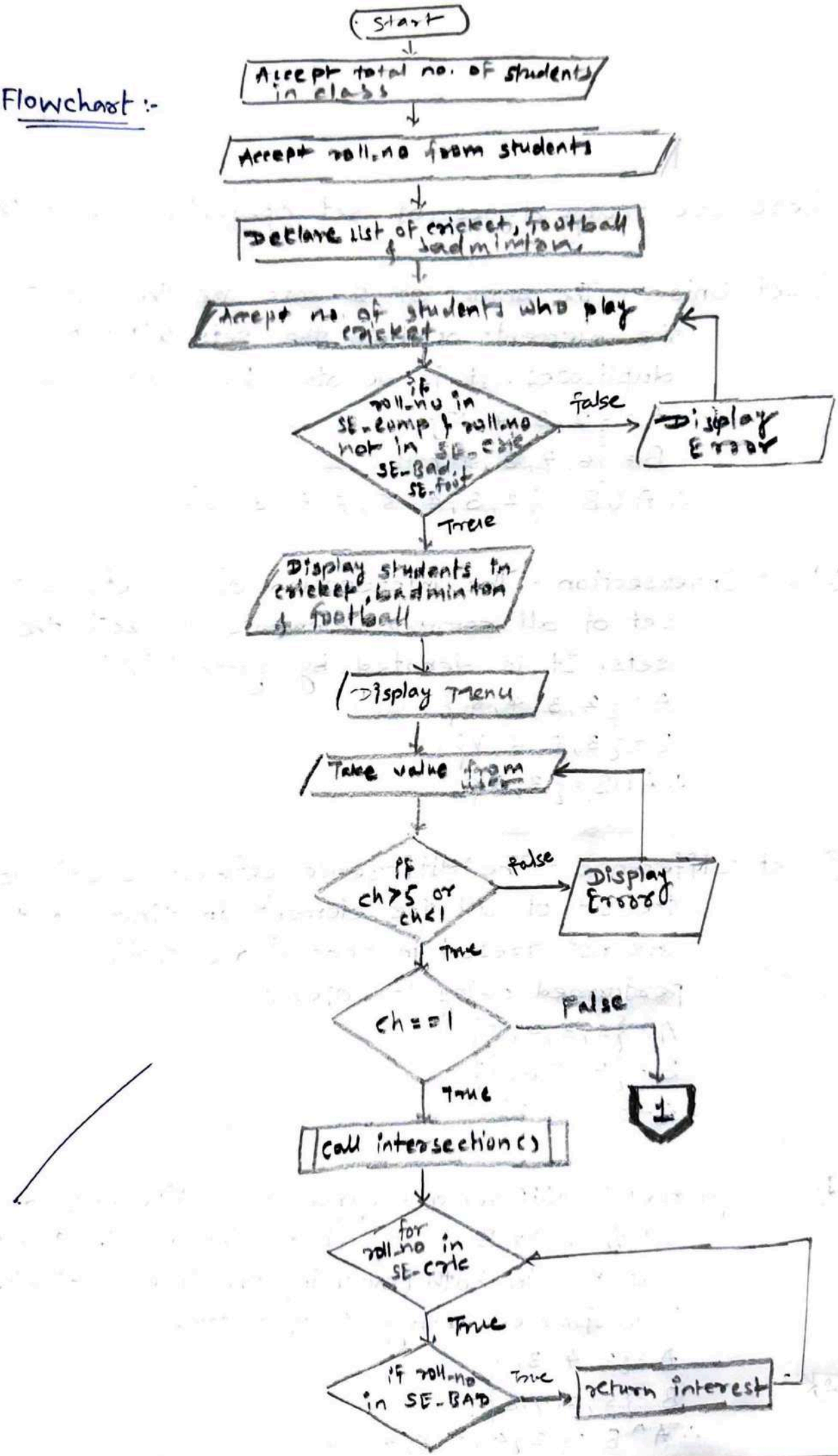
(d) Set Symmetric Difference - Symmetric difference of set A & set B is a set of elements in A & B but not in both (excluding the intersection). It is performed using ^ operator.

$A = \{2, 4, 3, 5\}$

$B = \{5, 6, 7, 8\}$

$\therefore A \hat{} B = \{2, 4, 3, 6, 7, 8\}$

* Flowchart :-


Start
Accept total no. of students in class
Accept roll_no from students
Declare list of cricket, football & badminton
Accept no of students who play cricket
if roll_no in SE_comp & roll_no not in SE_cric SE_Bad, SE_foot
false
Display Error
True
Display students in cricket, badminton & football
Display menu
Take value from user
if ch>5 or ch<1
false
Display Error
True
ch==1
False
1
True
Call intersection()
for roll_no in SE_cric
True
if roll_no in SE_BAD
True
return interest

DYP
Dr. D.Y. PATIL EDUCATIONAL COMPLEX AKURDI, PUNE

* Algorithm :-

Step 1 - Create three empty lists each representing Group A (Cricket), Group B (Badminton) & Group C (Football)

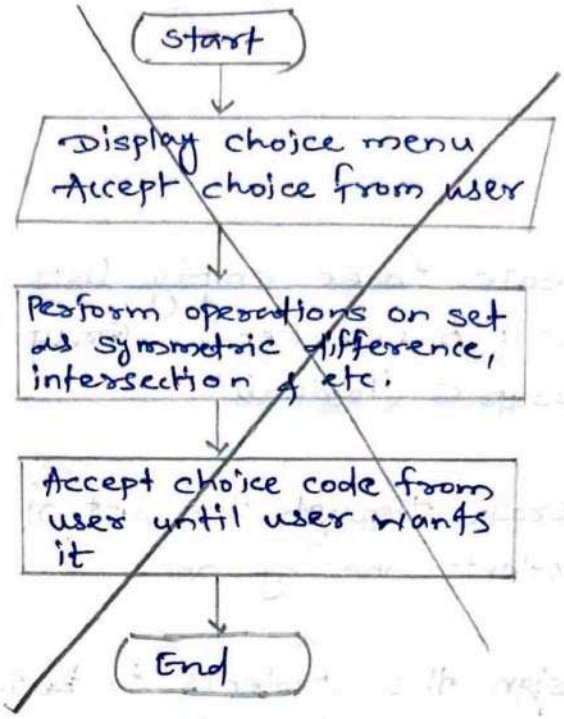
step 2 - Iterate through the list of 2nd year engineering students one by one.

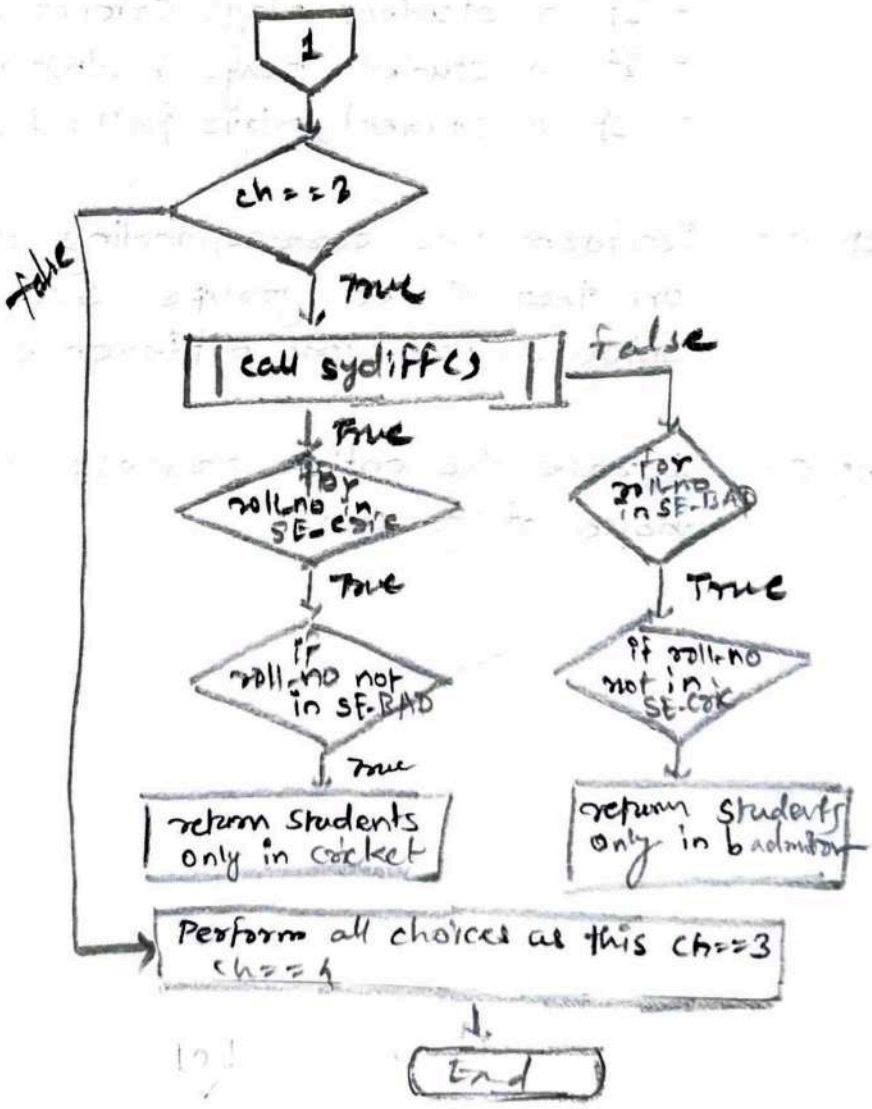
step 3 - Assign the students to the corresponding group based on their sport performance
- If a student plays Cricket, add to A
- If a student plays badminton, add to B
- If a student plays football, add to C

Step 4 - Perform the corresponding set operations on that three groups such as intersection, union, symmetric difference accordingly.

Step 5 - Iterate the entire process according to the choice of an user.

* Flowchart :-


Start
Display choice menu
Accept choice from user
Perform operations on set as symmetric difference, intersection of etc.
Accept choice code from user until user wants it
End


1
ch==3
false
True
call sydiff()
false
True
for rollno in SE-cric
for rollno in SE-BAD
True
True
if rollno not in SE-BAD
if rollno not in SE-cric
True
return students only in cricket
return students only in badminton
Perform all choices as this ch==3
ch==4
End

DYP
Dr D Y PATIL EDUCATIONAL COMPLEX AKURDI PUNE

* Test Cases :-

Sr.No.	Steps	Expected Output	Actual Output	Result.
1.	Accept data	Accept of Data will accepted by program from user.	"Data inputed successfully", message get displayed.	Pass.
2.	Union of two sets.	two sets will combined.	Union of two sets is successfull	Pass.
3.	Insection of 2 sets.	Common elements get grouped in this set.	Insection of 2 sets is successfull.	Pass.
4.	Symmetric Difference of two sets.	This set will consists the elements of both sets & except the intersection of both.	Set of Symmetric difference created successfully.	Pass.
5.	Exit from Program	Program will terminated.	"Thank for using our software"	Pass.

DYP
Dr. D.Y. PATIL EDUCATIONAL COMPLEX
AKURDI, PUNE

* Conclusion :-

After Performing this practical, we are able to performed set operations without using built-in function.

13/10/23

* Title :- Student Mark Analysis

* Objective :- The objective of the assignment is to create a python program that allows us to input marks scored by N students in a particular subject. The program will compute and display various statistics such as average score, highest marks, lowest score, the count of absent number, students & the marks with highest frequency.

* Problem statement :- Write a python program to : store marks stored marks scored in subject "Fundamental of data structure" by N students in the class.

Write functions to compute the following :-
(1) The Average score of the class
(2) Highest & lowest score of the class.
(3) Count of students who were absent for the test.
(4) Display marks with the highest frequency.

* Outcome :- By the end of this assignment, students will have developed a python program which is capable of analyzing student marks.

* Software Requirement :- i) Python Programming Environment.
ii) 32/64 -bit Linux (Ubuntu) Operating System.

* Hardware Requirement :- i) Any CPU with i3 processor.
ii) 1GB Ram or more.

DYP

Dr. D.Y. PATIL EDUCATIONAL COMPLEX AKURDI, PUNE

* Theory concept in brief :-

1) Lists :- Lists are a fundamental data structure in Python used to store a collection of items, in this case, student marks. They allow for easy indexing, iteration & modification of data elements.

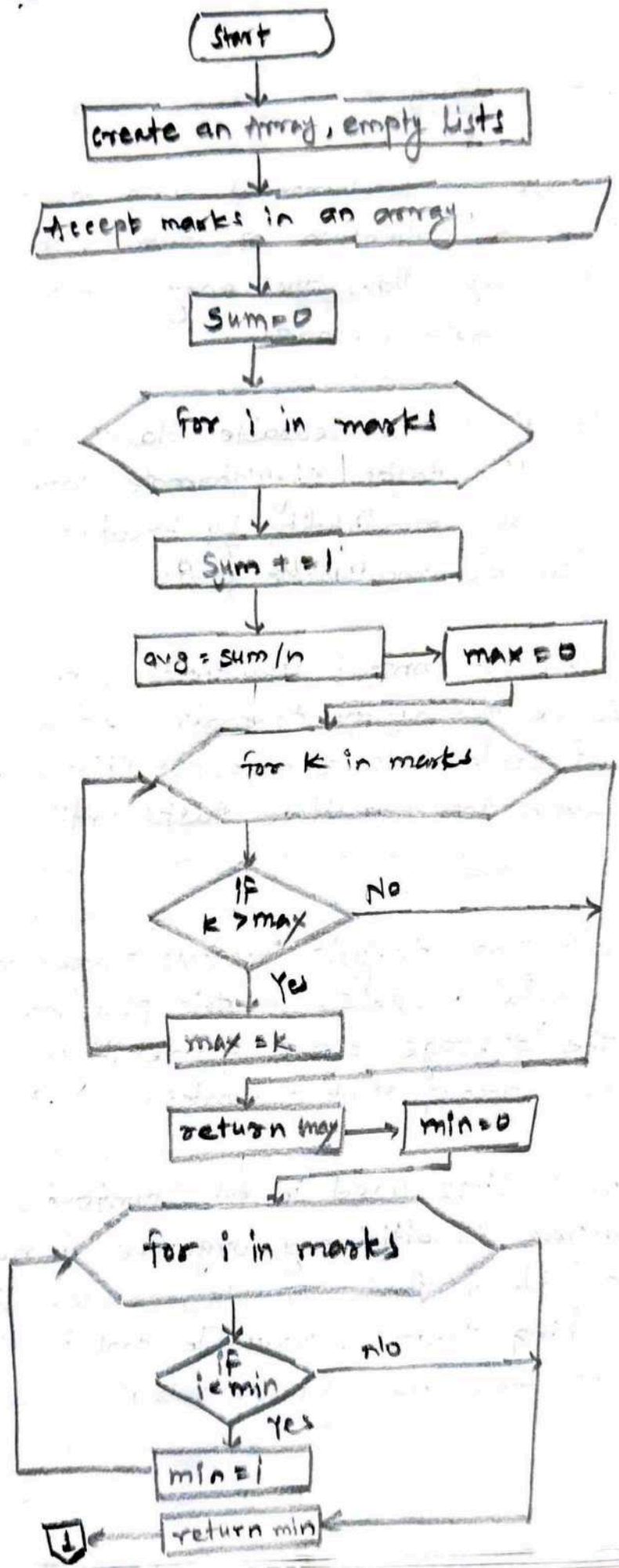
2) Functions :- Functions are reusable blocks of code that encapsulate specific tasks. They promote code modularity & acceptebility and readability by breaking down the program into smaller, managable parts.

3) Control structures :- Control structures like if statements & loops provide the ability to make decisions & iterate over data. If statements enable conditional execution, while loops allow for repetitive tasks until & condition in definition.

4) Data Analysis :- Data Analysis involves processing data of extract meaningful insights. In this practical, it include calculating the average score & identifying the highest & lowest scores among student marks.

5) Data Structure :- Lists used in this context, serve as a data structure to efficiency organize & manipulate data. They provide methods to add, remove & access elements, making them a valuable tool for handling collections of data like student marks.

* Flowchart :-


Start
create an array, empty lists
Accept marks in an array
Sum=0
for i in marks
Sum += i
avg = sum/n
max = 0
for k in marks
If k > max
No
Yes
max = k
return max
min=0
for i in marks
If i < min
no
yes
min = i
return min
1

DYP
Dr. D.Y. PATIL EDUCATIONAL COMPLEX AKURDI, PUNE

* Algorithm :-

Step 1 :- Start

step 2 :- Input the number of students (N).

step 3 :- Initialize an empty list 'student_marks'.

step 4 :- Loop N Times
a) Input the marks for each student.
b) Validate input & append marks to 'student_marks'.

step 5 :- Calculate the average score using to 'avg' function.

step 6 :- Find the highest score using the 'max' function.

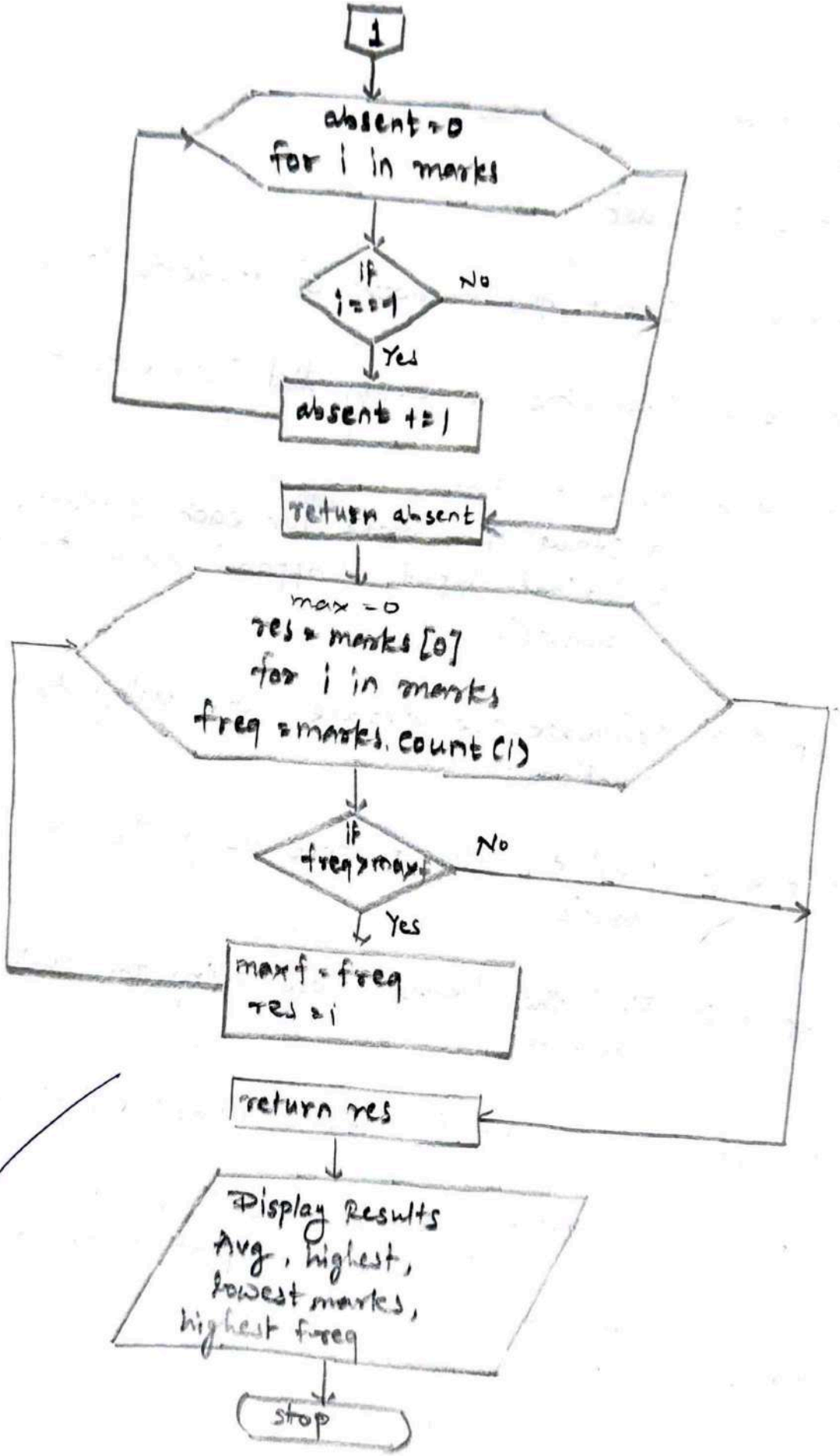
step 7 :- Find the lowest score using the 'min' function

step 8 :- Count absent students using 'absect' function.

step 9 :- Find the mark with the highest frequency using the 'mark_with_highest_frequency' function.

step 10 :- Display results.

step 11 :- End


1
absent = 0
for i in marks
if i == 1
No
Yes
absent += 1
return absent
max = 0
res = marks[0]
for i in marks
freq = marks.count(i)
if freq > max
No
Yes
maxf = freq
res = i
return res
Display Results
Avg, highest,
lowest marks,
highest freq
stop

24/10/23

DYP
Dr. D.Y. PATIL EDUCATIONAL COMPLEX AKURDI, PUNE

* Test Cases :-

Sr. No.	Steps	Expected o/p	Actual o/p	Result
1.	Accept marks of 5 students as n=5	Marks of 5 students should accepted.	Marks of 5 students accepted.	Pass.
2.	Average marks of class	Average marks should displayed.	Average marks of class displayed.	Pass.
3.	Compare marks & find highest & lowest marks.	Highest & lowest marks should displayed.	Highest & lowest marks displayed	Pass.
4.	Find no. of absent students.	No. of absent students should displayed.	Absent students figure get displayed.	Pass.
5.	Find the marks which have highest frequency.	Highest frequency marks should displayed.	Highest frequency marks get displayed.	Pass.

* Conclusion :- This assignment has provided a hands on experience in developing a python program to handle & analyse student marks. This program can be used for data analysis in future.

24/10/23

* Title :- To implement various function to perform various operations on string.

* Objective :- To understand the basic techniques & strategies of algorithm analysis to be perform various operations on string without using built-in functions.

* Problem Statement :- Write a python program to compute following operations on string :-
  i) To display word with longest length.
  ii) To determine frequency of occurance of particular character in string.
  iii) To check whether given string is palindrome or not.
  iv) To display index of appearence of sub string.
  v) To count the occurance of each word in the string.

* Outcome :- i) Implementation of concepts of arrays like inserting, indexing.
  ii) Getting Identical with concept of functions & concept related to it.

* Hardware & Software Requirements :-
  1) 32/64 bit linux (Ubuntu) OS.
  2) Python Programming enviornment.
  3) Text Editor.
  4) Disk space, 4GB+ RAM.
  5) Processor - i3+ GEN

DYP
Dr. D.Y. PATIL EDUCATIONAL COMPLEX AKURDI, PUNE

* Theory :-

- String - String is a sequence of characters string is a special datatype which holds the characters or alphabets as well as numbers as a list of characters.

- Function - A Function is a block of code which only runs when it is called by a parameter. A block of code is seperately defined under a name which can be called multiple times as required without redefined it every time.

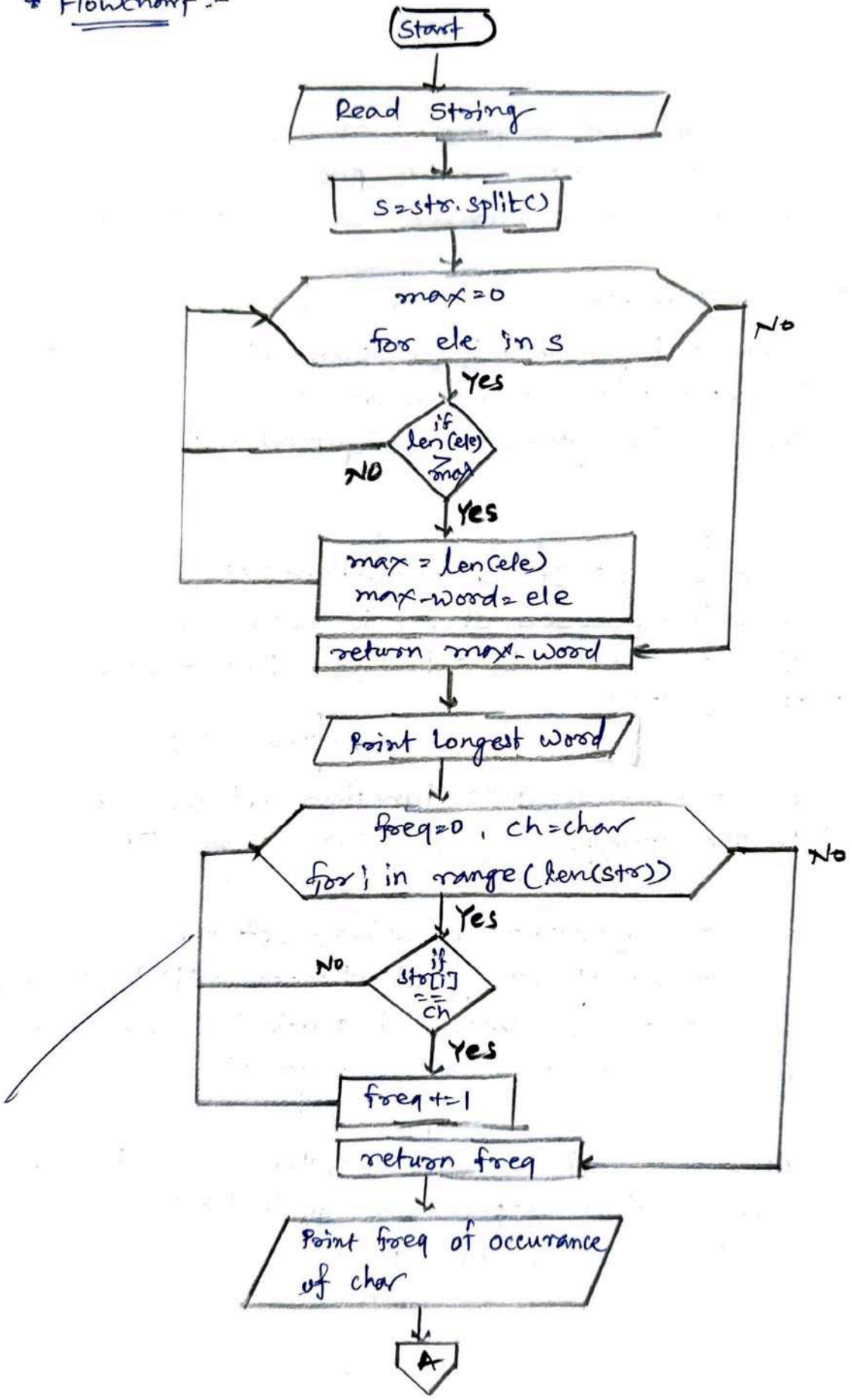
- Palindrome string - If all characters of a string are reversed & if reversed string is equal to original string then the string is called as "Palindrome String."
Eg. MOM , LEVEL

- Legth of String - length() function returns the longest word in the string.

- Frequency of characters in string - Let, the string is "This is string." In this frequency of character 'i' is 3. Thus, we can calculate frequency of particular character by defining a function.

- First appearence of String - Let, the string is "This is String", here first appearence of substring is at index 5. Thus we can find first.

* Flowchart :-


Start
Read String
S=str.split()
max=0
for ele in S
No
Yes
if len(ele) > max
No
Yes
max = len(ele)
max_word = ele
return max_word
Print Longest word
freq=0 , ch=char
for i in range(len(str))
No
Yes
if str[i] == ch
No
Yes
freq += 1
return freq
Print freq of occurance of char
A

DYP
Dr. D.Y. PATIL EDUCATIONAL COMPLEX AKURDI, PUNE

* Algorithm :-

(1) Start
(2) Read String.
(3) Split the string and store in variables.
(4) Create a function to find longest word

```
max = 0
for ele in s:
    if len(ele) > max:
        max = len(ele)
        max_word = ele
return max_word
```

(5) Print logest word.
(6) Create a function to find frequency of particular character.

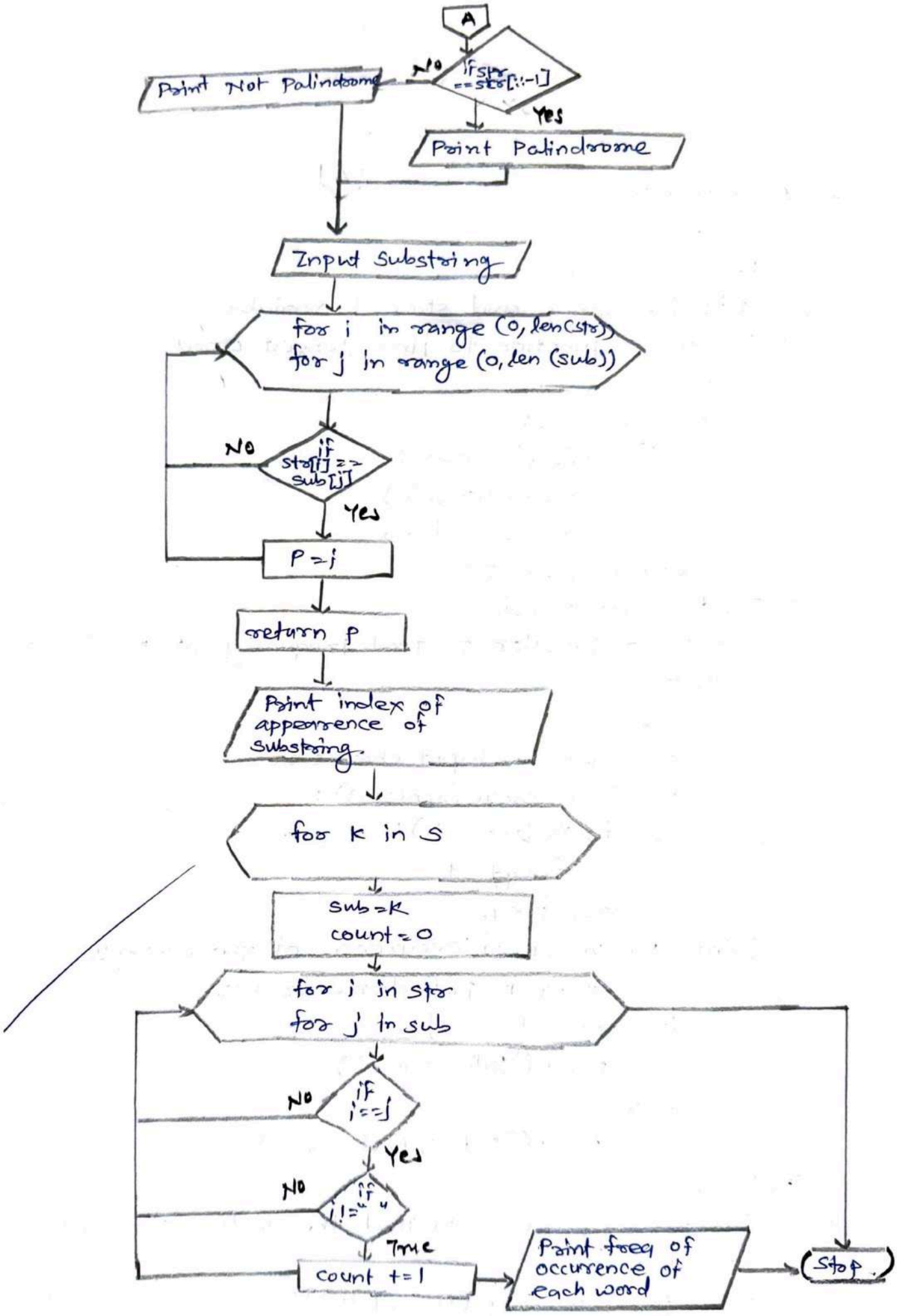
```
freq = 0
ch = char i.e. input character.
for i in range(len(str)):
    if (str[i] == ch):
        freq += 1
return freq.
```

(7) Print frequency of occurence of the character.
(8) Check a string is palindrome or not.

```
if str == str[::-1]:
    print("Palindrome")
else:
    print("Not Palindrome")
```

(9) Read Substring.
(10) Create a function to find the index of appearence of substring

```
for i in range(0, len(str)):
    for j in range(0, len(substr)):
```


A
if str == str[::-1]
NO
Yes
Print Not Palindrome
Print Palindrome
Input Substring
for i in range (0, len(str))
for j in range (0, len (sub))
if str[i] == sub[j]
NO
Yes
P = j
return p
Print index of appearence of substring.
for k in S
sub = k
count = 0
for i in str
for j in sub
if i == j
NO
Yes
if i != " "
NO
True
count += 1
Print freq of occurence of each word
Stop

```
        if(str[i] == substr[j]):
            p=i
        else:
            break
    return p
```

(11) Print the index of appearence of substring.

(12) Create a function to count occurance of each word in string :-

```
for k in s:
    sub = k
    count=0
    for i in str:
        for j in sub:
            if(i==j):
                if(i!=" ")
                    count+=1
                else
                    break
```

(13) Print frequency of occurance of each word.

(14) Stop.

* Conclusion :- Thus we have studied the string operation performed successfully.

DYP
Dr. D.Y. PATIL EDUCATIONAL COMPLEX AKURDI, PUNE

* Test Cases :-

Description	Expected Output	Actual Output	Result.
1) Display longest word.	Display student	Word "student" displayed.	Pass
2) Display frequency of occurance of particular character.	ch = s display "1"	"1" displayed & shows 's' appeared once in string.	Pass
3) To check string is palindrome or not.	Display "Not Palindrome"	Displayed palindrome.	Fail
4) Display index of appearence of substring.	substring: student display "7"	"7" displayed and show sub-string student is present of index 7	Pass.

* Conclusion :- Practical of string operation performed successfully.

11/11/23

4a

* Title :- To find the search key using linear search & sentinel search.

* Objectives :- To understand linear & sentinel search.

* Problem Statement :- Write a python program to store roll number of student in array attented training program in random order. Write function for searching whether particular student attended training program or not.

* Outcomes :- Students will be able to use linear & sentinel search technique.

* Hardware & software requirements :-
1) 32/64 bit linux
2) Python 3.10.7
3) storage - 500GB+
4) RAM - 8GB+
5) Text editor.

* Theory :-
Searching is the process of finding the required information in computer memory.

• Linear Search :-
Linear search is defined as sequential search algorithm that starts at one end & goes through each element of list until the desired element is found, else the search continues till end of data.

Time Complexity = O(n)
Space Complexity = O(1)

- Sentinel search :-

It is a type of linear search where the number of comparisons is reduced as compared to a traditional linear search.

Time Complexity = O(n)
Space Complexity = O(n)

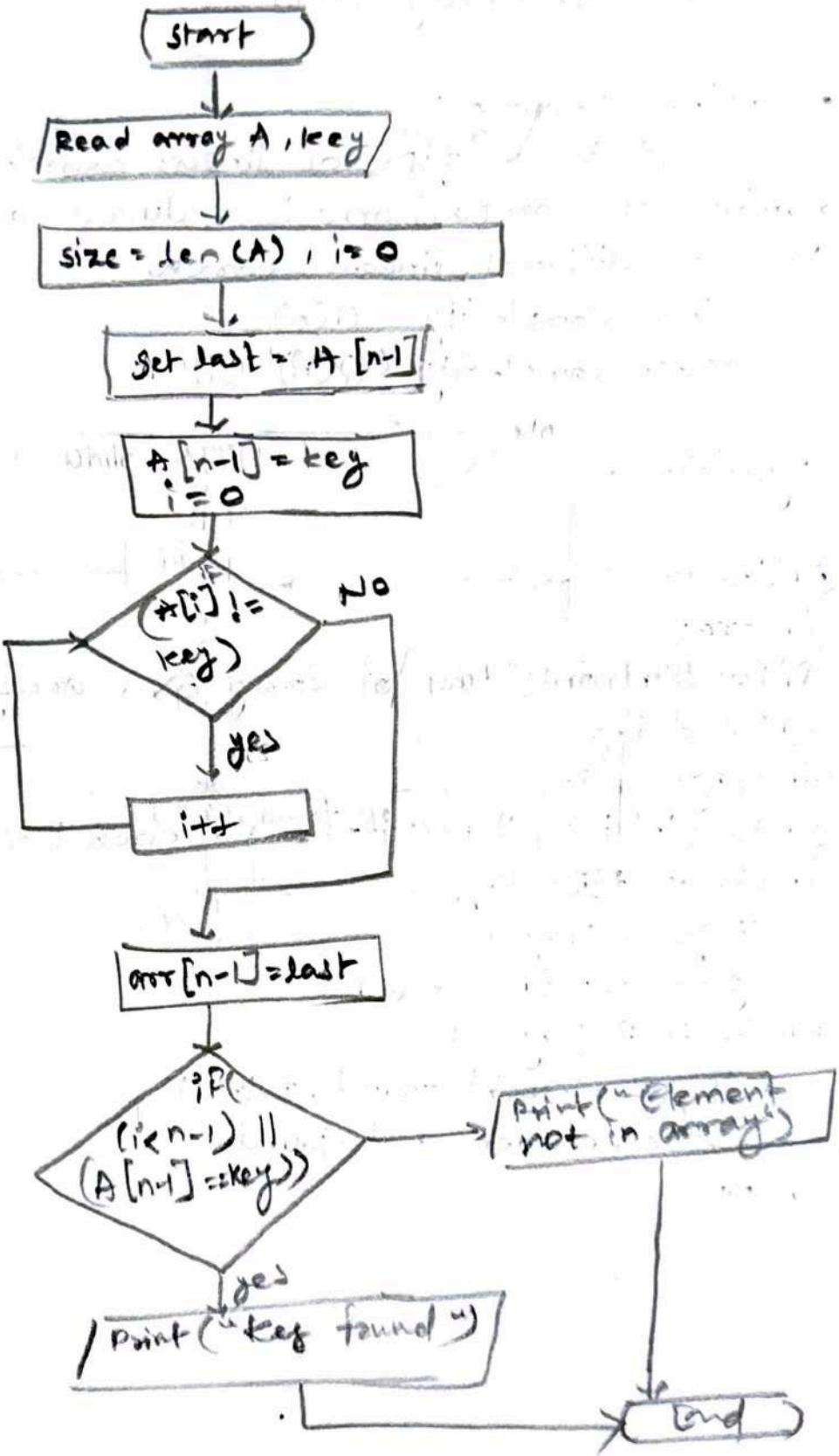
* Algorithm :-

(A) Linear Search -

(1) Start
(2) Read n, the size of array A & array elements
(3) Read key.
(4) flag = 0, i = 0
(5) for (int i=0; i<n-1; i++) .... for each i value do step (6)
(6) check A[i] == key
if yes, flag = 1
goto step (7) else (5)
(7) check if flag == 1
if yes, element found, print i
else, element not found.
(8) stop.

* Flowchart :-

A) Sentinel Search :-



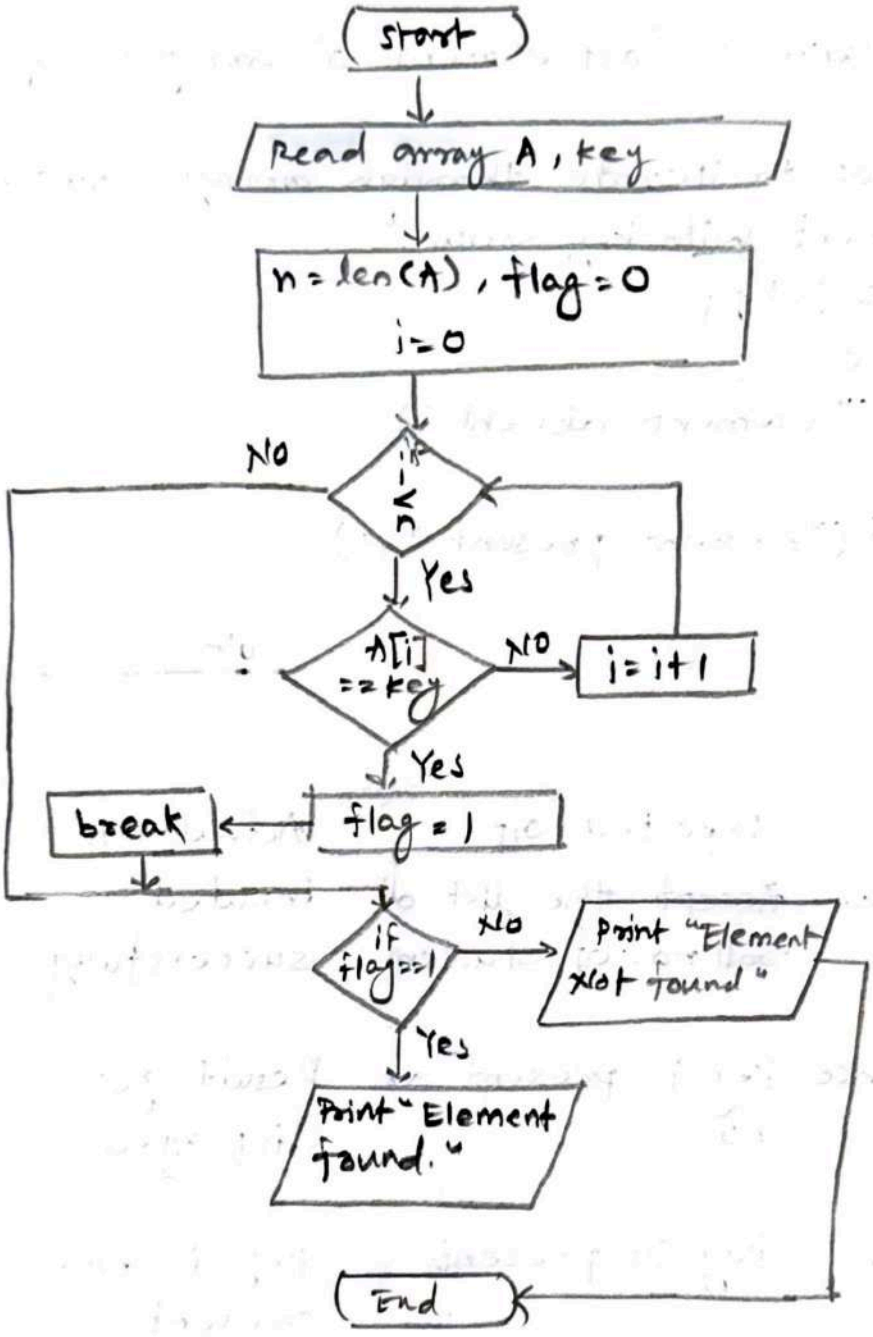
(B) Sentinel search -

(1) Start
(2) Set the value of last element of array to key.
(3) set i = 0
(4) Use a loop to iterate through array, comparing each element with key value.
(5) if element != key
(6) if i == size
        print("element absent")
    else
        print("element present", i)
(7) Stop.

* Test Cases :-

	Description	Expected O/p	Actual O/p	Result
1)	Accept data.	Accept the list of roll no. of student.	inputed successfully.	pass.
2)	check presence of key.	key is present or not.	Result get displayed.	pass.
3)	check / enter key.	key is present	key is not present	fail
4)	Display index of key is found.	Display the index of key if present.	Result get displayed successfully.	Pass.

B) Linear Search :-



* Conclusion :-

Implementation of linear search & sentinel search is done successfully.

DYP
Dr D.Y.PATIL EDUCATIONAL COMPLEX AKURDI, PUNE

4b

* Title :- To find the search key using binary search & fibonacci search.

* Objective :- Understand implementation of binary & fibonacci search.

* Problem Statement :- Write a program to store roll no. of students in array attended training program in sorted order. Write a function for searching whether particular student attended training program or not using binary & fibonacci search.

* Outcome :- Student will understand how to implement binary & fibonacci search

* Hardware & Software Requirements :-
1) 32/64 bit linux
2) Python enviornment.
3) Text editor.
4) RAM - 8GB+
5) Processor - i3+
6) Disk space - 256GB+

* Theory :-
Searching is the process of finding required information from computer memory.

- Binary search :-
It is a searching algorithm used in sorted array by repeatedly dividing search interval in half.

Time Complexity = O(log n)
Space Complexity = O(1)

- Fibonacci Search :-

The algorithm uses fibonacci numbers to search for an element in sorted order. This algorithm is similar to binary search.

Time Complexity = O(log n)
Space Complexity = O(1)

* Algorithm :-

A) Binary Search :-

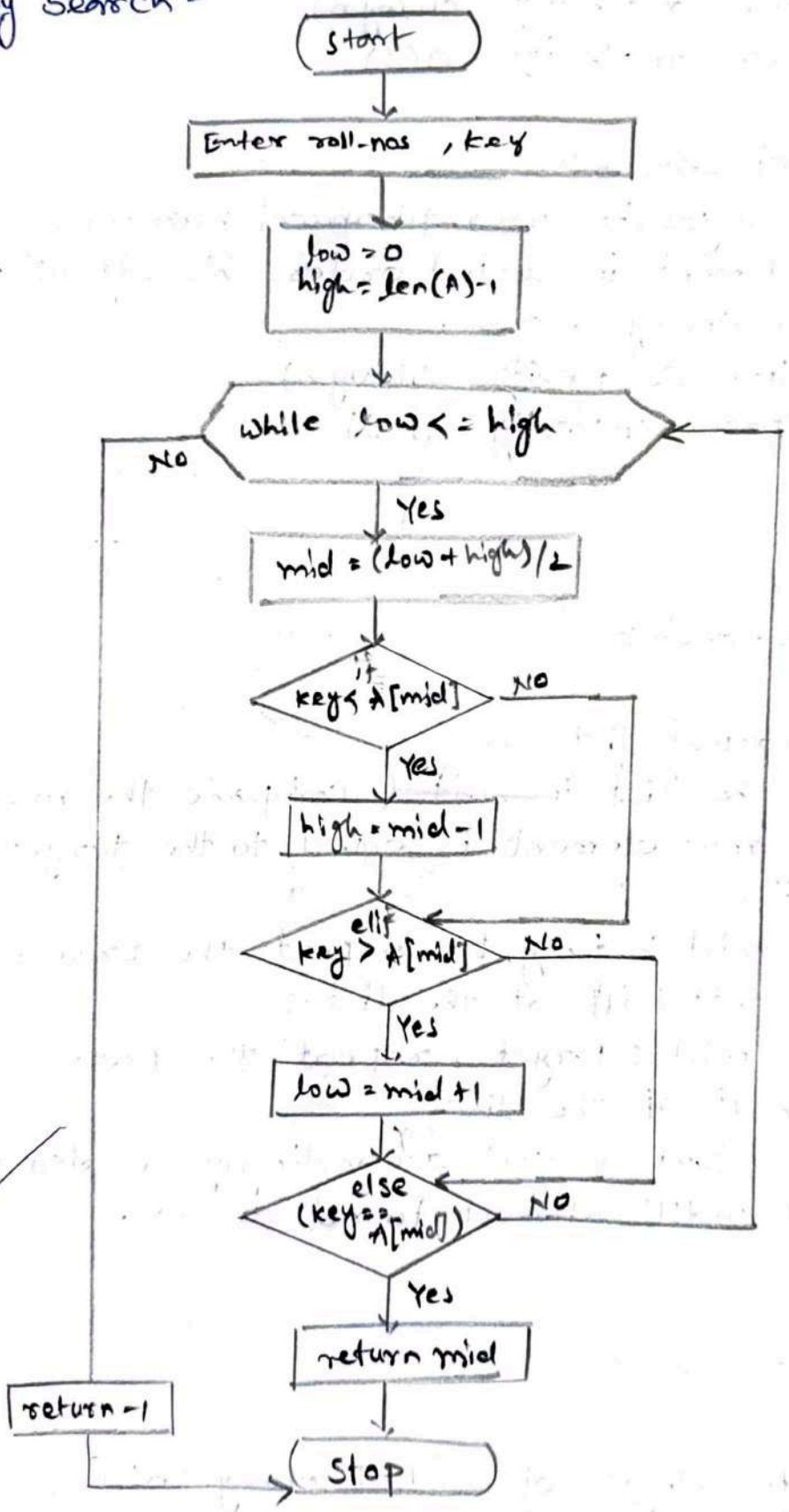
1. Start.
2. Input sorted list.
3. Divide the list in half & compare the mid to target
4. If the mid element is equal to the target, you've found it.
5. If the mid > target, repeat the process of on the left half of the list.
6. If the mid < target, repeat the process on the right half of the list.
7. Update start & end accordingly & also mid too.
8. Repeat until element found or s>e
9. Stop.

B) Fibonacci Search :-

1. Start
2. Calculate length of sorted array 'n'.
3. Initialize Fibonacci members :

* Flowchart :-

A) Binary Search -



'fibminus2 = 0', 'fibminus = 1', 'fibM = fibminus1 + fibminus2'

(3) find the smallest fibonacci no. $\geq n$

(4) offset = -1

(5) While fibM > 1 :

(a) Calculate i to be probed.

(b) If arr[i] == target, return 'i'

(c) If arr[i] < target, reduce the search range & adjust fibonacci nos.

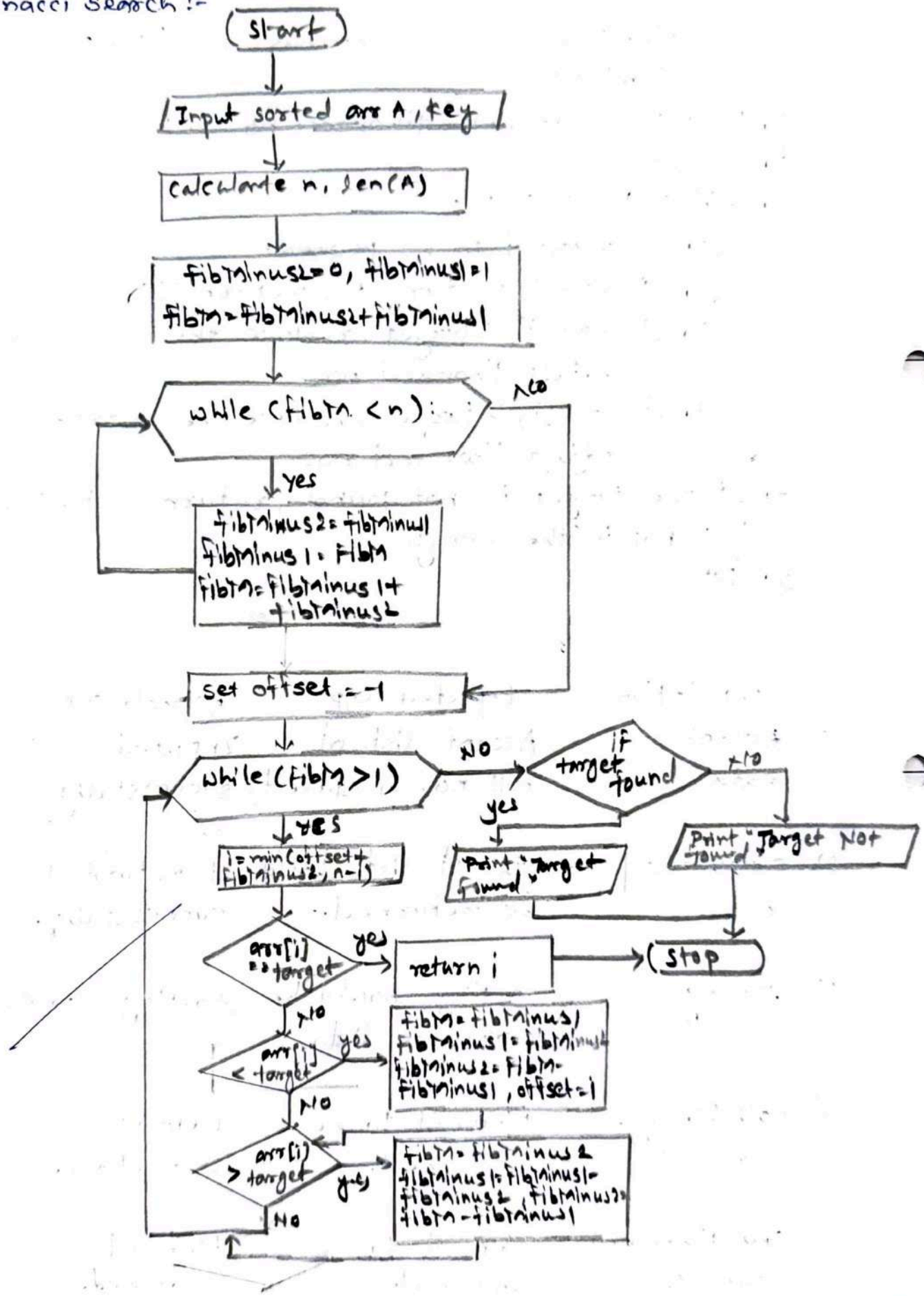
(d) If arr[i] > target, reduce the search range & adjust fibonacci nos.

(6) If the target is not found, return -1 to indicate it's not in the array.

(7) Stop.

	Description	Expected o/p	Actual o/p	Result
1)	Accept data.	Accept list of roll no. of student.	Inputed successfully.	Pass
2)	calculate n.	Size of list should be returned.	It returned successfully.	Pass
3)	sorting	Sorting should be done on list.	Sorting done.	Pass
4)	call binary-search()	Element to be searched.	Element searched.	Pass
5)	call fibonacci-search()	Element to be searched.	Element searched.	Pass

B) Fibonacci search :-



DYP
Dr. D.Y. PATIL EDUCATIONAL COMPLEX
AKURDI, PUNE

* Conclusion :-

Thus, implementation of searching algorithms- fibonacci search & binary search done successfully.

8/11/23

* Title :- To implement selection & bubble sort using python.

* Problem statement :- To understand various functions of python. Write a program to store first year percentage of students in array. Write function for sorting array.

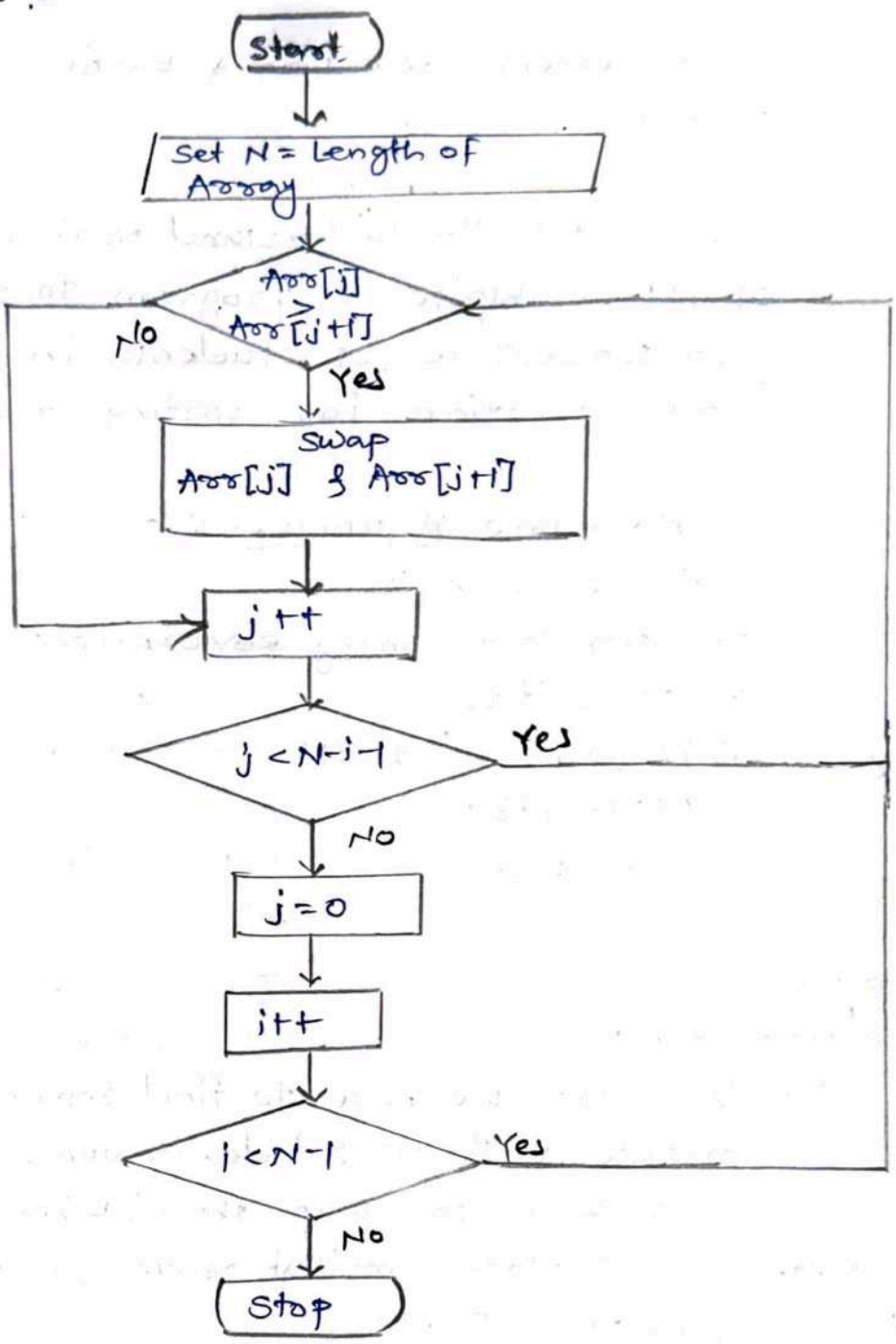
* Software & Hardware Requirement :-
1) 64 bit linux OS
2) Python Programming enviornment
3) Text Editor.
4) Processor - i3+ GEN
5) RAM - 4GB+
6) Disk space.

* Theory :-
• Selection sort :-
In this sort, we have to find minimum value in array & compute with first index value, if we found minimum, we have to swap them. When one iteration completed we'll place smallest value on first index.
Time Complexity - $O(n^2)$

• Bubble sort :-
In this sorting algorithm, comparison take place. In which each pair of adjacent elements are supported if they are not in order.
Time Complexity - $O(n^2)$

* Flowchart :-

1) Bubble Sort :-



DYP
Dr. D.Y. PATIL EDUCATIONAL COMPLEX AKURDI, PUNE

* Algorithm :-

- selection sort :-

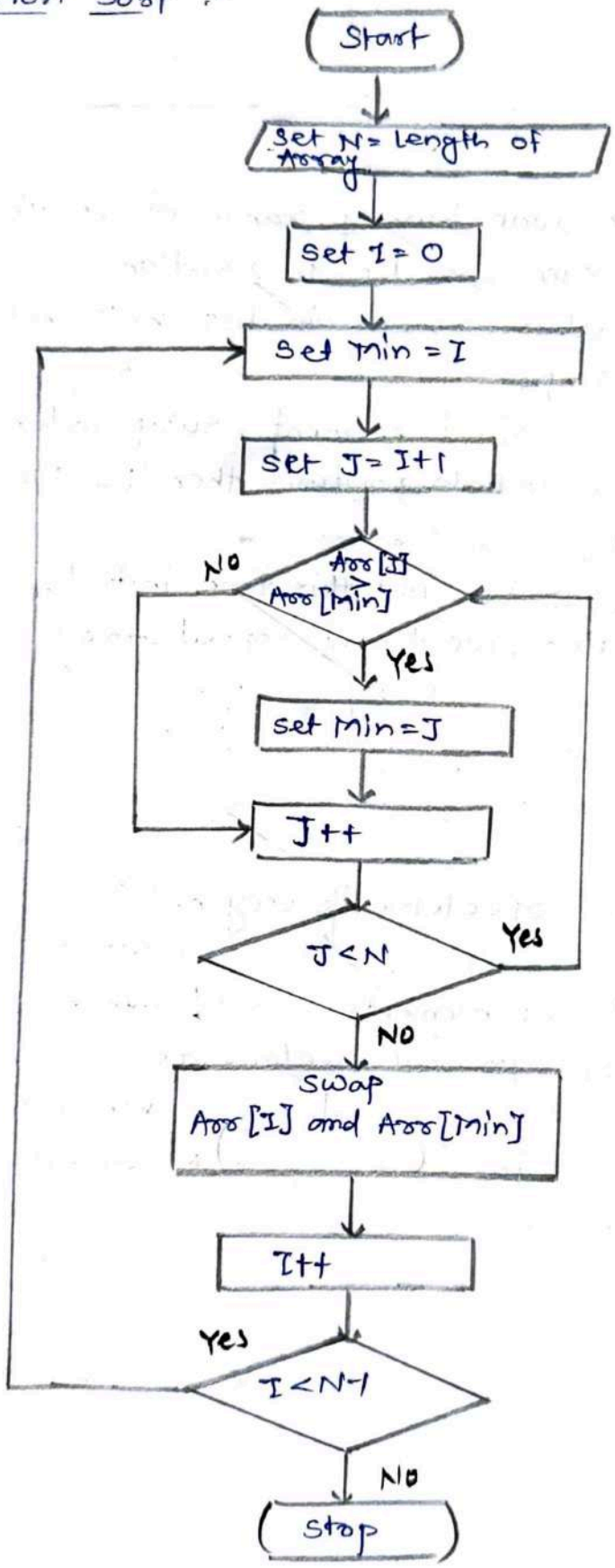
(1) Start
(2) Start first for loop having range of length of list.
(3) Initialize minimum variable to location 0
(4) Look for smallest element in list with help of second for loop.
(5) After finding smallest element, swap index no. of smallest value to 0^{th} position then it's position is fixed in list.
(6) Repeat step (2), (3) & (4) but this time initialize min variable to next position, we'll get sorted array.
(7) Stop.

- Bubble sort :-

(1) Start
(2) Read total no. of elements say n.
(3) set i=0
(4) Compare adjucent elements
(5) Repeat step (4) for all n-elements.
(6) i++
(7) Repeat (4) & (5) for i≤n
(8) print sorted list.
(9) Stop.

* Flowchart :-

2) Selection Sort :-



DYP
Dr. D.Y. PATIL EDUCATIONAL COMPLEX
AKURDI, PUNE

* Test Cases :-

	Description	Expected output	Actual Output	Result
1)	Accept Data	Percentage of students in list	percentage accepted	Pass
2)	selection sort	Selection sort for given list	Percentages list get sorted.	Pass
3)	Top 5 percentages	Give top 5 percentages from list.	Top 5 percentages get displayed.	Pass

* Conclusion :-

We have successfully implemented selection sort & bubble sort.

11111123

DYP
Dr. D.Y. PATIL EDUCATIONAL COMPLEX AKURDI, PUNE

* Title :- To implement Quick sort using python.

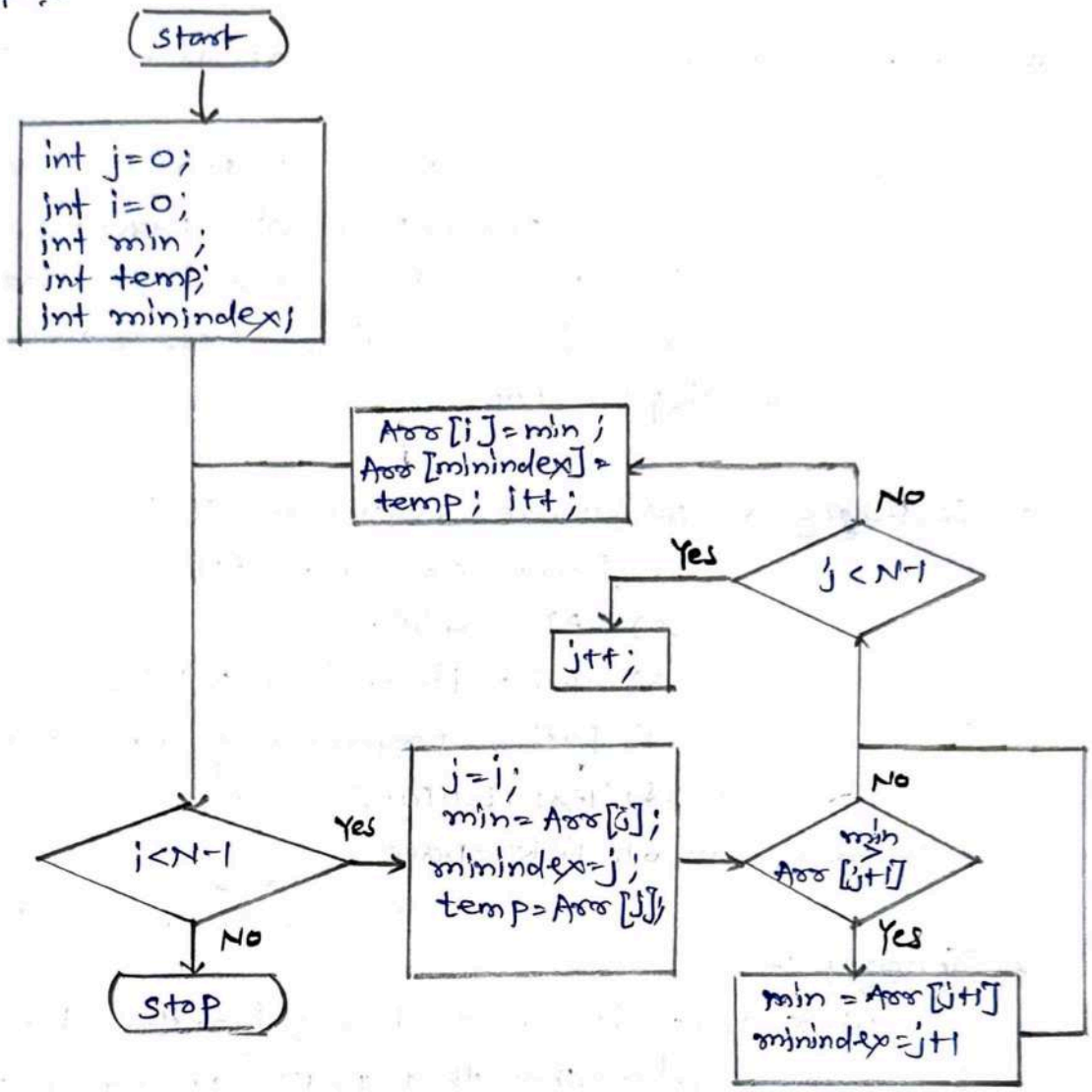
* Problem Statement :- Write a python program to store first year percentage of students in array. Write function for sorting array of floating point numbers in ascending order using quick sort & display top 5 scores.

* Software & Hardware Requirements :-
1) Processor - i3+ GEN
2) RAM - 4GB+
3) 64 bit Linux (Ubunto) OS
4) Python programming enviornment.
5) Text Editor
6) Disk space.

* Theory :-
Quick sort is a sorting algorithm based on divide & conquer algorithm that picks an element as a pivot & partitions the given array around the picked pivot by placing the pivot in it's correct position in sorted array.

It works on basis of partition. The target of partition is to place the pivot element by choosing it to correct position in sorted array & put smallest element to left of pivot and all greater elements to right of pivot.

* Flowchart :-


Start
int j=0;
int i=0;
int min;
int temp;
int minindex;
Arr[i] = min;
Arr[minindex] = temp; i++;
NO
Yes
j < N-1
j++;
i<N-1
Yes
j=i;
min = Arr[i];
minindex=j;
temp = Arr[j];
NO
min > Arr[j+1]
Yes
min = Arr[j+1]
minindex=j+1
NO
Stop

DYP
Dr. D.Y. PATIL EDUCATIONAL COMPLEX AKURDI, PUNE

* Algorithm :-

(1) Start
(2) Consider first element of list as pivot element.
(3) Define 2 variables i, j. set i & j to 1st & last element
(4) Increament i until list[i] > pivot then stop.
(5) Decrement j until list[j] < pivot then stop.
(6) if i < j then interchange list[i] & list[j].
(7) Repeat step (3), (4) & (5) until i > j
(8) Exchange pivot with list[j] elements.
(9) Stop.

For the top 5 scores, Print last 5 elements of list.

* Test Cases :-

	Description	Expected output	Actual Output	Result.
1)	Accept data	Accept percentage of students.	Accept the percentage of Students.	Pass.
2)	Quick sort	List to be sorted	List get sorted	Pass.
3)	shows valid percentage.	Doesn't showing percentage greater than 100.	Does not showed percentage greater than 100.	Pass.
4)	Top 5 percentage.	Top 5 percentage should displayed.	Top 5 scores get displayed.	Pass.

* Conclusion :- We have successfully implemented Quick sort & displayed top 5 scores.

11/11/23

DYP

Dr. D.Y. PATIL EDUCATIONAL COMPLEX, AKURDI, PUNE

(8)

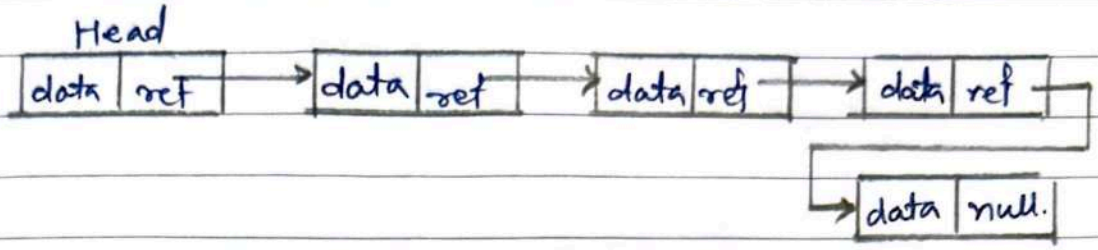
* Title :- Write c++ program to maintain club member's information using singly linked list.

* Objectives :- 1) To understand the concept & features of singly linked list data structure.
2) To main club member's information by performing different operations like add, delete, concate & total count of nodes on singly linked list.

* Hardware & Software Requirements :-
1) Open source linux derivation.
2) G++/Gcc compiler
3) Text editor
4) i3+ processor
5) RAM 4GB+
6) Disk space - 500 GB+

* Theory :-
• Linked List :-

A Linked list is a sequence of data structures, which are connected via links. A linked list contains various nodes where each node contains data field and a reference (link) to the next node of the list.
Linked list can be visualized as a chain of nodes, where every node points to the next node.



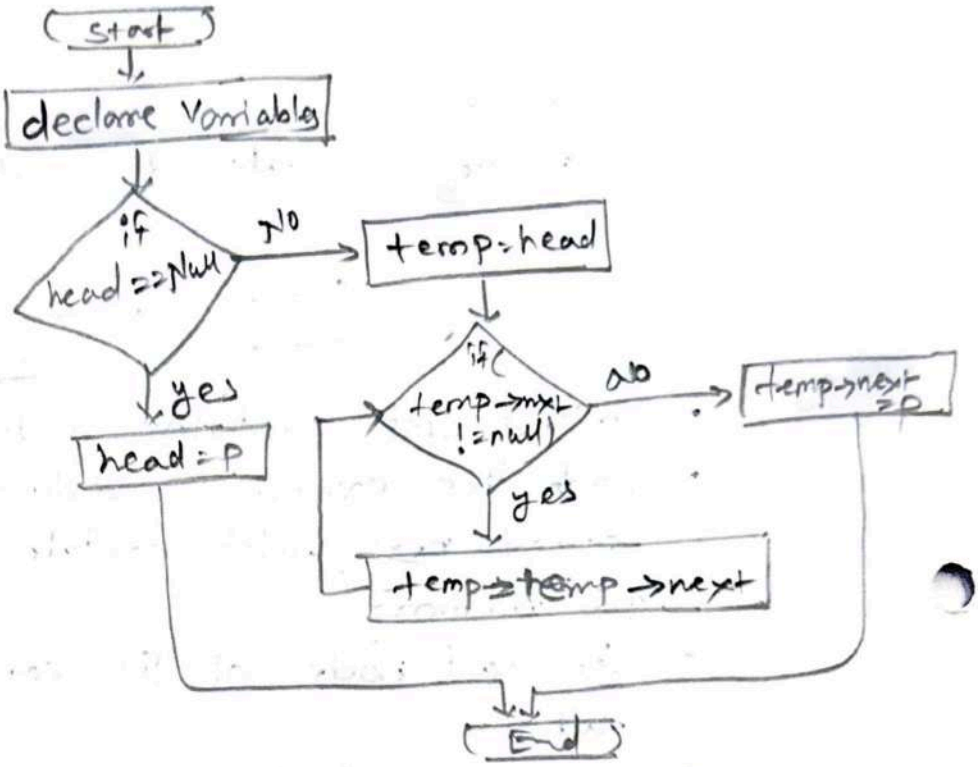
- Linked list contains a link element called first.
- Each link carries a data field & a link field called next which contains the address of the the memory.
- The last node of list contains pointer to the null.
- Types of Linked list :-
  1. Singly linked list
  2. Circular linked list
  3. Doubly linked list.

Singly linked list :-

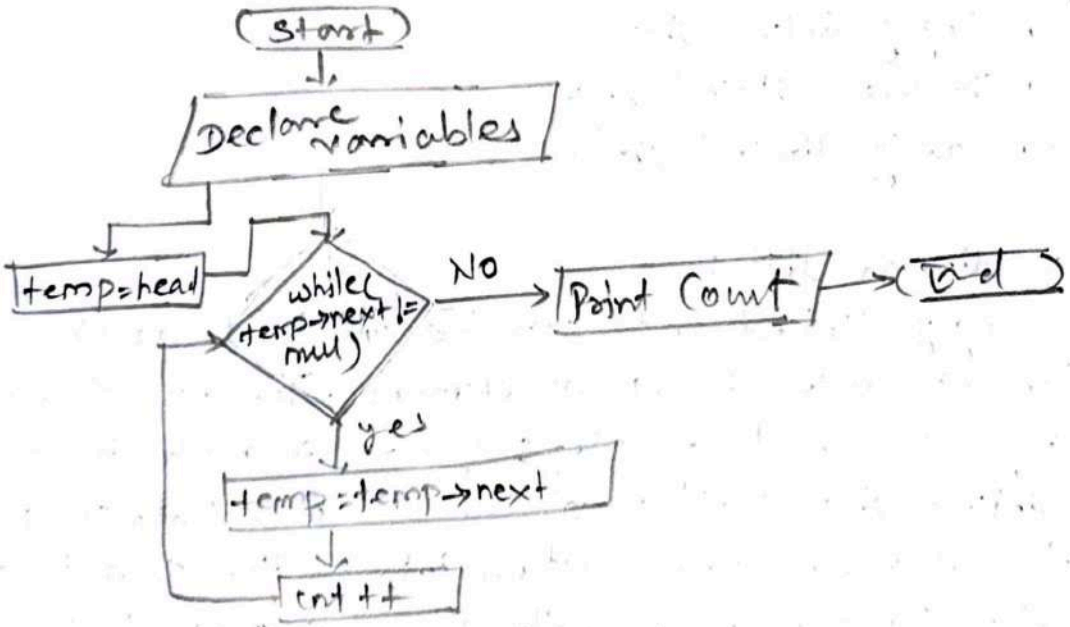
Singly linked list can be defined as the collection of ordered set of elements. The no. of element according to need of the program. A node in the singly linked list consist of two parts : Data part & part of the node stores actual information that is to be represented by the node while the link part of the address of its successor.

- Operations on Singly linked list :-
  a) Linked list insertion.
  b) searching
  c) length calculation.
  d) Traversing.
  e) Deleting node.

Flowchart :- for addtnember c) :-



for count c) :-



f) Reverse a linked list.
g) Creating linked list.

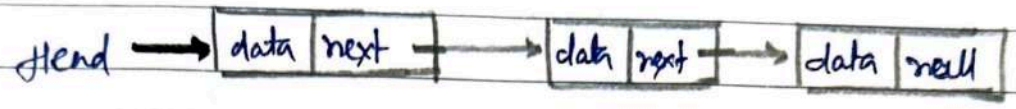


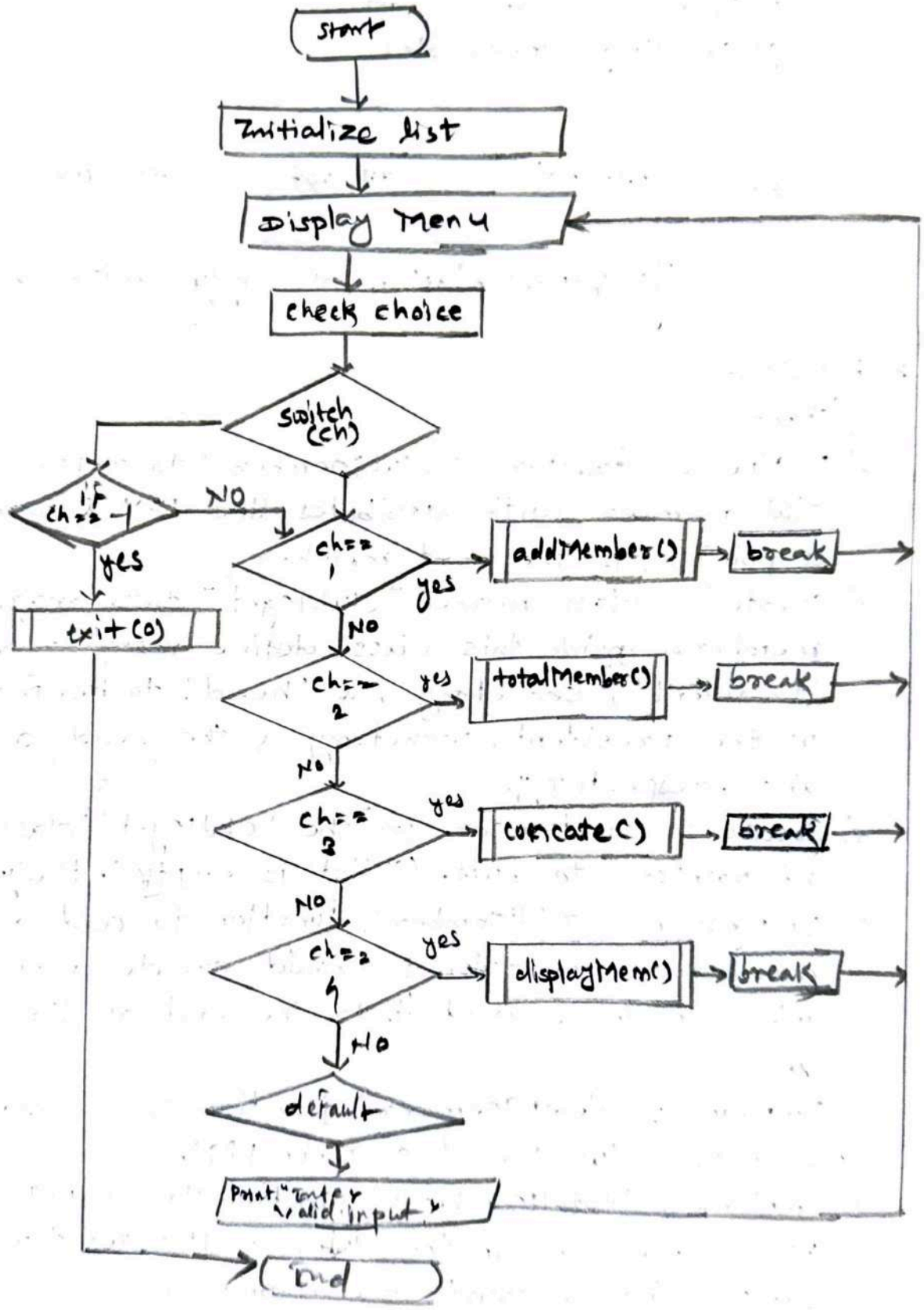
fig. Representation of Singly linked list.

* Algorithm :-

1) Start
2) Define a structure "ClubMember" to represent club member with attributes like PRN & name. Include a pointer to the next member.
3) Create a class named "ClubMgnt" to manage the club members. Inside this class, define member variables. 'president', 'secretary', & 'head' to keep track of the president, secretary & the head of the linked list, respectively.
4) Create a constructor for the 'ClubMgnt' class. Initialize all pointers to null. (∵ list is empty initially).
5) Implement 'addMember' function to add a new member to the club. It should create a new 'ClubMember' node & add it to the end of the linked list.
6) Implement 'deleteMember' function to remove a member from the club based on their PRN.
7) Implement 'totalMembers' function to compute the total no. of members in the club. It iterates through the linked list & counts the members.

* Flowchart :-

for main() -



8) Implement the 'displayMem' function to display the information of all club members. It traverses the list & prints the PRN & name of each member.

9) In the 'main' method :-
   a) Create object of 'ClubMgnt' class.
   b) Use the 'addMember' method to add members to the club.
   c) Display the member's information using the 'displayMem' method.
   d) Calculate & display the total no. of members using 'totalMembers' method.

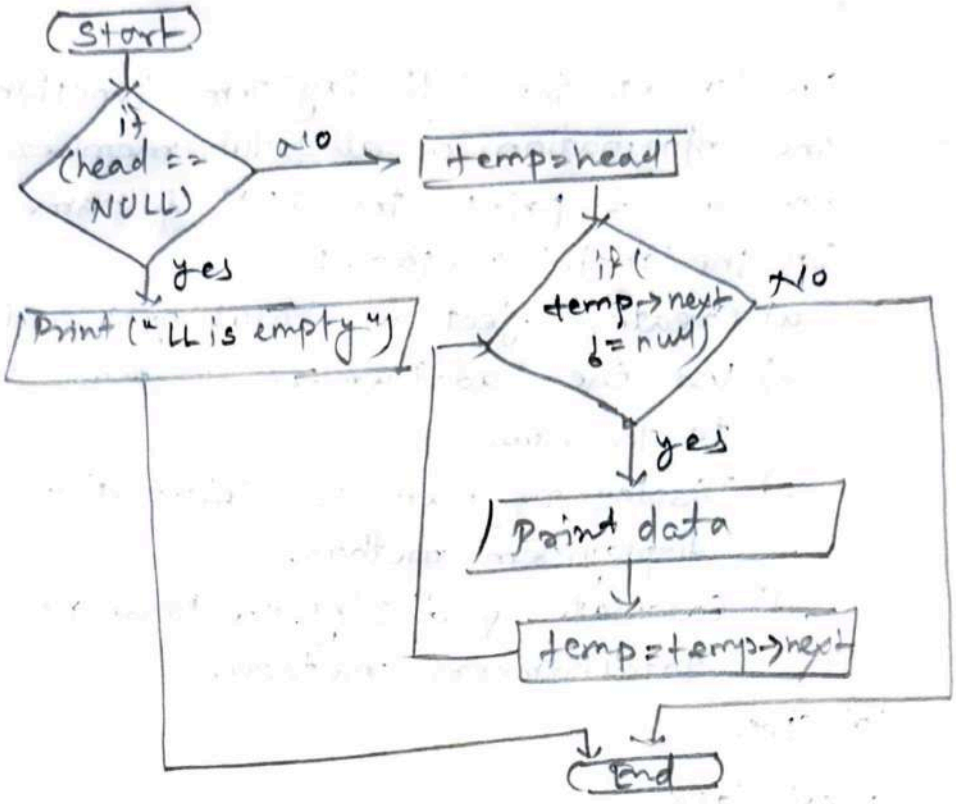
10) Stop.

Test Cases

* :-

	Description	Expected o/p	Actual o/p	Result
1)	New node addition.	New node should be inserted.	New node is inserted.	Pass.
2)	Deletion of node	Specified node should be deleted.	Specified node is deleted.	Pass.
3)	Display of member.	Specified node details should get displayed.	Details get displayed.	pass.
4)	Total no. of members.	Total should be displayed.	Total no. get displayed.	Pass.
5)	Traversal of Linked list.	Pointer should get traversed list.	Pointer get traversed to whole linked list.	Pass

for display mem() :-



DYP
Dr. D.Y. PATIL EDUCATIONAL COMPLEX
AKURDI, PUNE

* Conclusion :-

In conclusion, the integral study & implementation approach of singly linked list in real world.

* Title :- Write a C++ program to implement ticket booking system for Cinemax theatre.

* Objectives :- To understand the implementation of Circular linked list like display, insert & delete, etc.

* Problem Statement :- The ticket booking system for Cinemax theatre has to be implemented using C++ program. There are 10 rows & 7 seats in a each row. Doubly Circular linked list has to be maintened to keep track of free seats at rows. Assume some random booking to start with. Use array to store pointers (Head Pointer) to each row. On demand -
a) The list of available seats to be displayed.
b) The seats are to be booked.
c) The booking can be cancelled.

* Hardware & Software Requirement :-
1) Text Editor
2) C++ compiler
3) RAM - 4GB+
4) Disk space - 256GB+
5) Linux OS
6) Processor - i3+ Gen.

* Theory :-
• Circular Linked List -

In Singly linked list, every node points to its next node in the sequence and the last node points

to Null.

But, in circular linked list, every node points to it's next node in the sequence but, the last node points to the first node in the list.

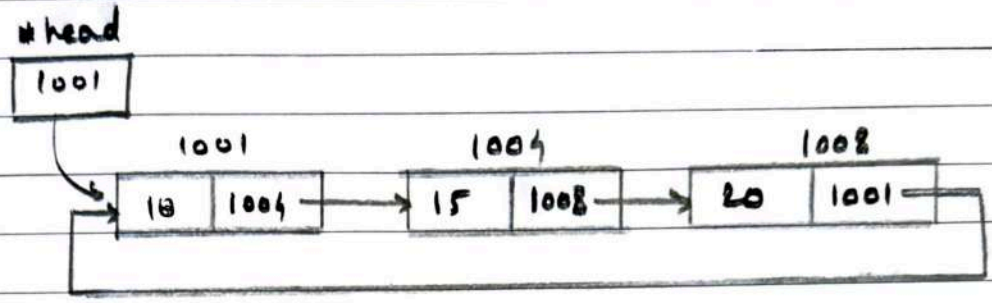


Fig. Representation of Circular linked list

- Doubly Circular linked list :- Here, the next pointer of the last node points to the first node & the previous pointer of the first node points to the last node in both directions.

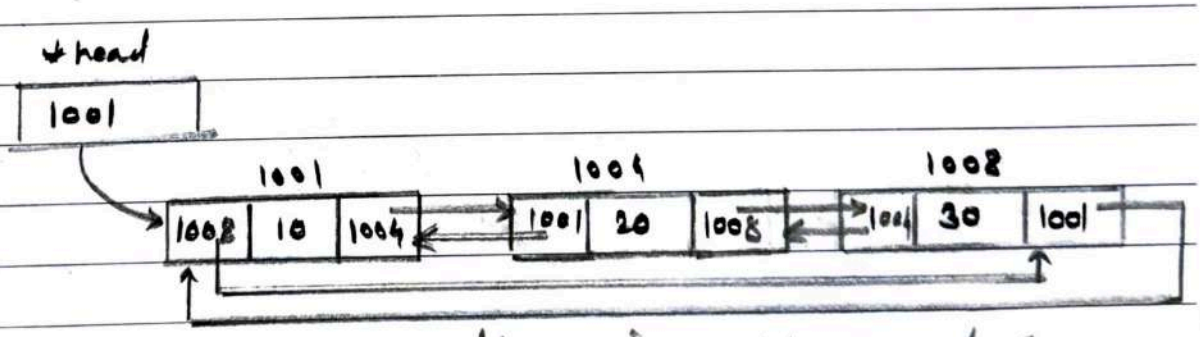
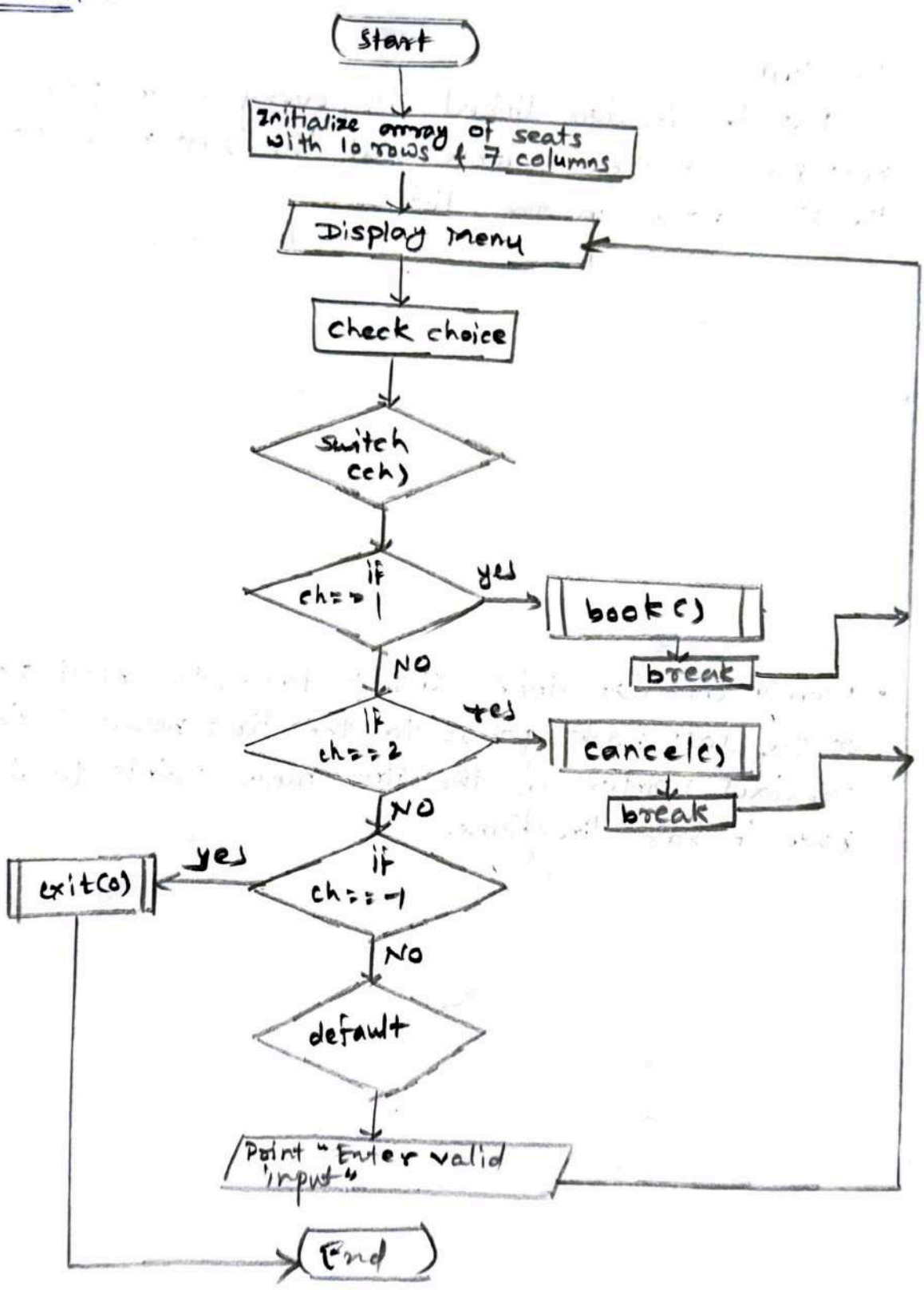


Fig. Representation of Doubly Circular linked list

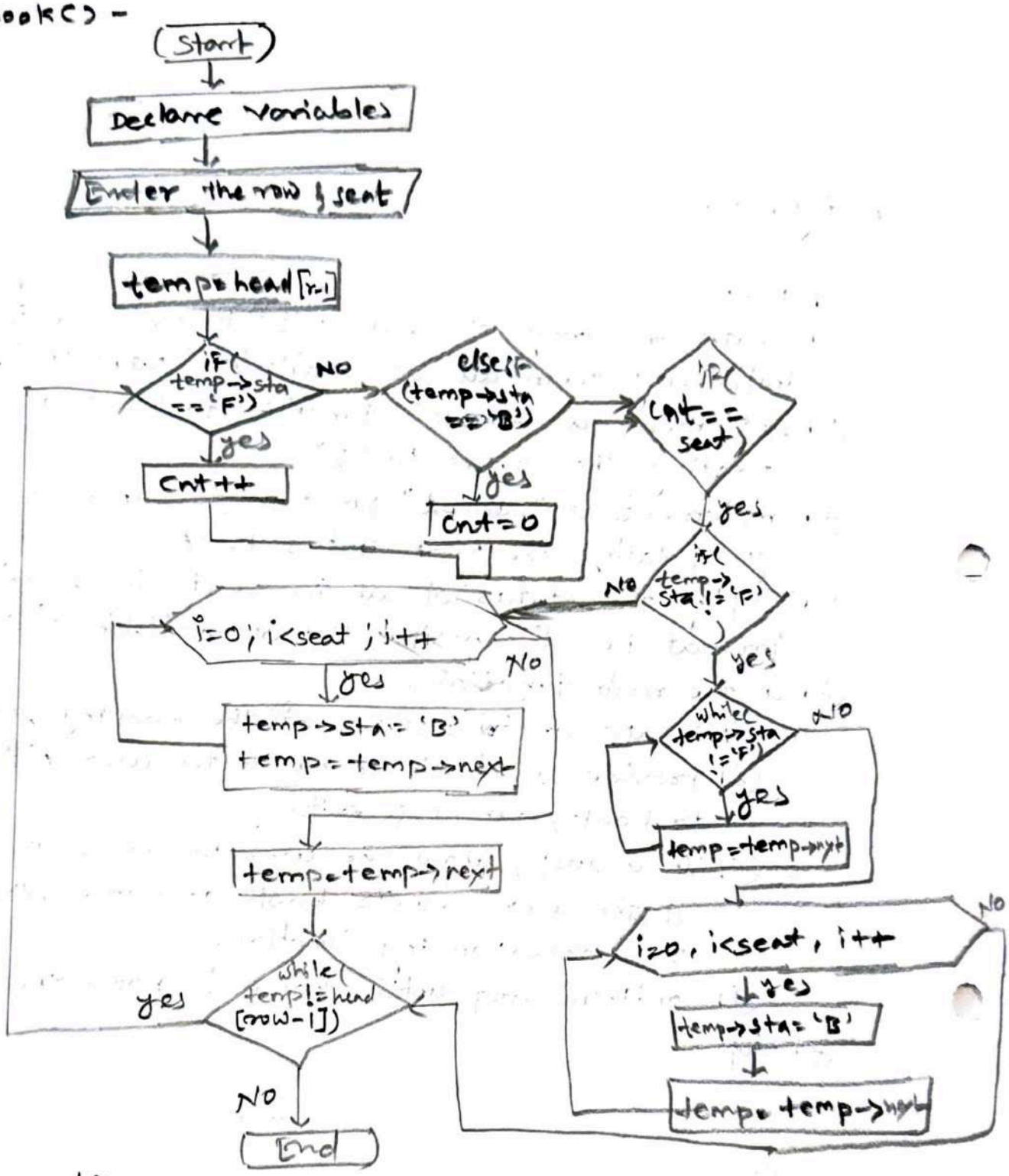
* Flowchart :- for main() -



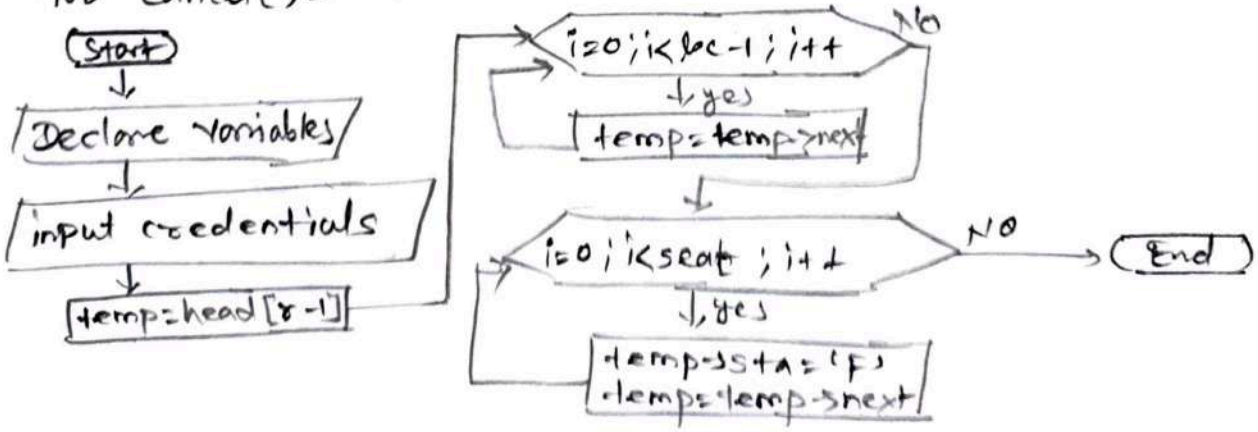
* Algorithm :-

(1) Start
(2) Create an array of seats with 10 rows & 7 columns initializing them all as available. (Status "A").
(3) Implement a "book" function to book a seat by updating its status to "B"
(4) Implement a "cancel" function to cancel a booking by updating the seat status to "A"
(5) Display the status of all the seats in a grid-like format i.e. display the list of available seats.
(6) In the main function,
    (a) Create an instance of the booking system.
    (b) Provide a simple menu to the user with options to book, cancel & exit.
    (c) In a loop, allow the user to choose an option. If the user selects book or cancel, execute the corresponding function.
    (d) Continue loop until choice is not equal to exit.
(7) Stop.

for book() -



for cancel() -



DYP
Dr. D.Y. PATIL EDUCATIONAL COMPLEX AKURDI, PUNE

* Test Cases :-

Description	Expected o/p	Actual o/p	Result
1) Check whether the grid for availability of seats is displayed or not.	Available seats should be displayed.	Availability of seats is displayed.	Pass
2) Check whether a seat is book or can be booked.	Available seats should be available for booking.	Seats can be booked.	Pass
3) Check whether the booking can be cancelled or not.	A booked ticket should be able to be cancelled.	Ticket can be cancelled	Pass
4) Check whether the remaining seats a deallocated from memory or not.	The leftover seats should be cancelled from the list.	Remaining seats aren't getting deleted / cancelled from the list.	Pass.

* Conclusion :-

In this practical, we have successfully implemented ticket booking system program with doubly circular linked list concept along with all of it's operations implementation.

* Title :- Write a c++ program to check whether given string is palindrome or not using stack using array.

* Objectives :- ① To study stack using array
② To check whether the given string is palindrome or not.

* Problem Statement :- A palindrome is a string of character that's the same forward & backward. Typically, punctuation, capitalization & spaces are ignored. Write C++ program with functions -
a) To print original string followed by reversed string using stack
b) To check whether given string is palindrome or not.

* Hardware & Software Requirement :-
1) Text Editor
2) C++ compiler.
3) Processor - i3+ Gen
4) RAM - 8GB+
5) Disk space - 256GB+
6) Linux OS.

* Theory :-

• Stack -

Stack is a LIFO (Last In First Out) structure. It is an ordered list of the same type of elements. A stack is a linear list where all insertions & deletions are permitted at only one end of the list.

DYP
Dr. D.Y. PATIL EDUCATIONAL COMPLEX AKURDI, PUNE

- Stack using Array -

Stack is an LIFO structure. Stack can be represented by using array. A 1D array can be used to hold elements of stack. Another variable "top" is used to keep track of the index of the top most element.

- Operations on stack -

1) push()
2) pop()
3) top()
4) size()
5) empty()
6) clear()

- Palindrome String :-

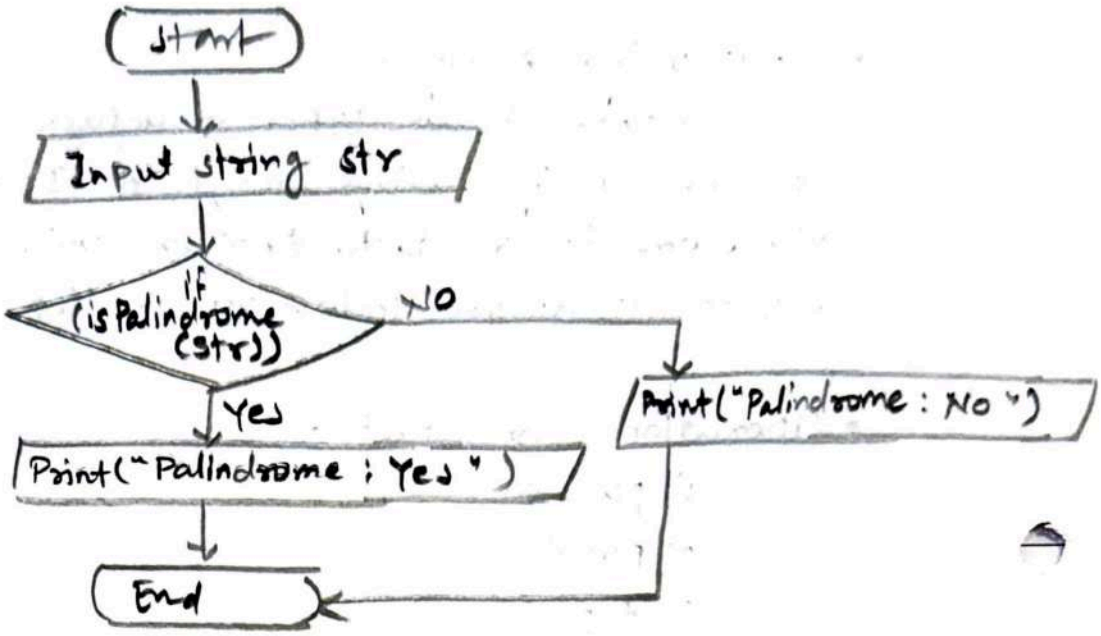
A palindrome is a word, sentence, or number that reads same from left to right & vice versa.

* Algorithm :-

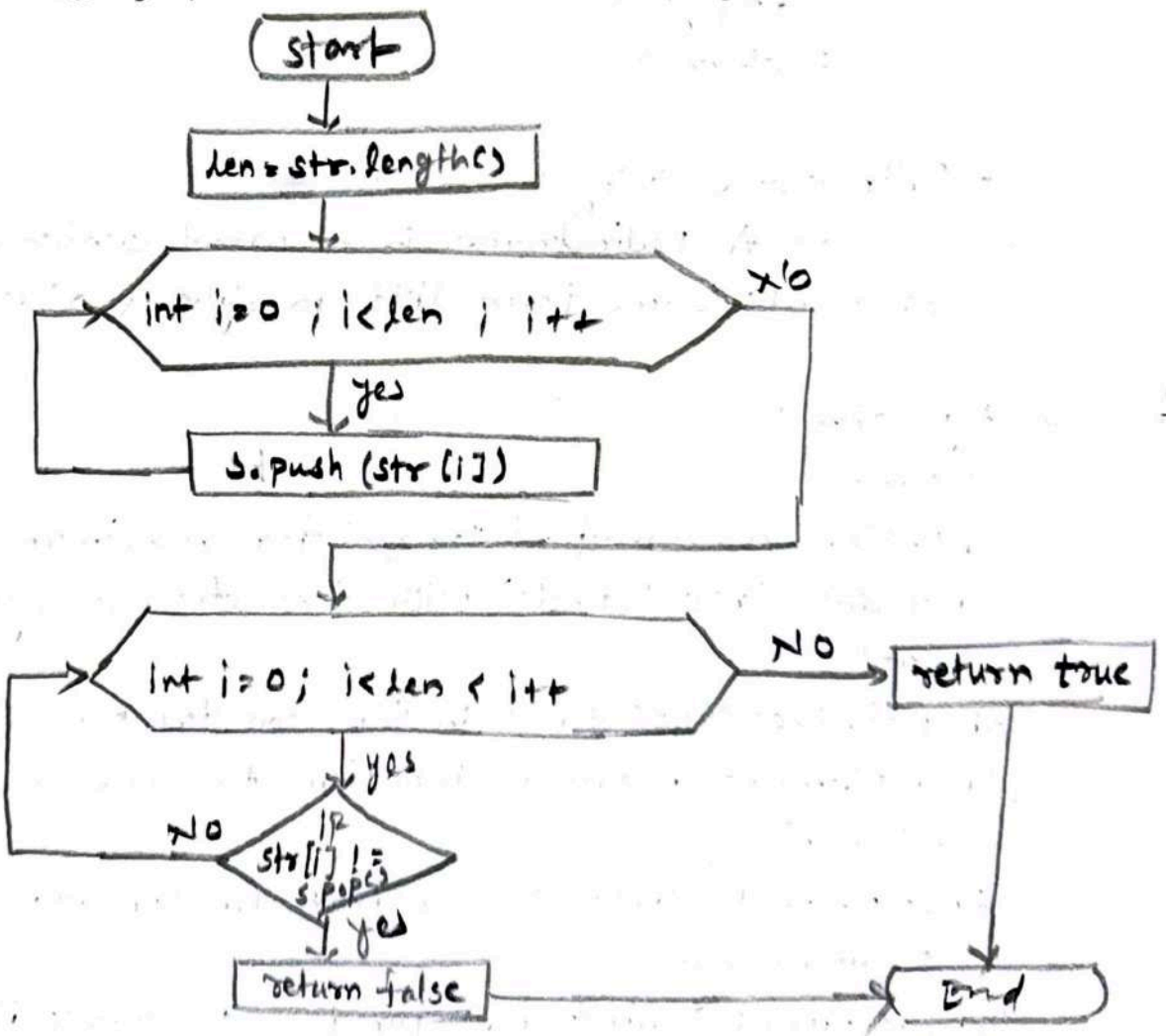
(1) Start
(2) Define a constant "max" for maximum stack size.
(3) Create class "stack" with character array A & integer 'top'
(4) Initialize 'top' to -1 in the constructor
(5) Implement "reverse" function to reverse the stack element
(6) Implement "convert" function to convert input string to lowercase.
(7) Implement "palindrome" function to check if the string in the stack is a palindrome or not.

* Flowchart :-

Main() :-



isPalindrome() :-



(8) Also, implement functions like "is_empty()", "is_full()" for stack operations.

(9) In main() function;

(a) Create instance a stack class.

(b) Input a string from the user.

(c) Convert the string to lowercase & push each character onto the stack.

(d) check if the string is a palindrome & print the result.

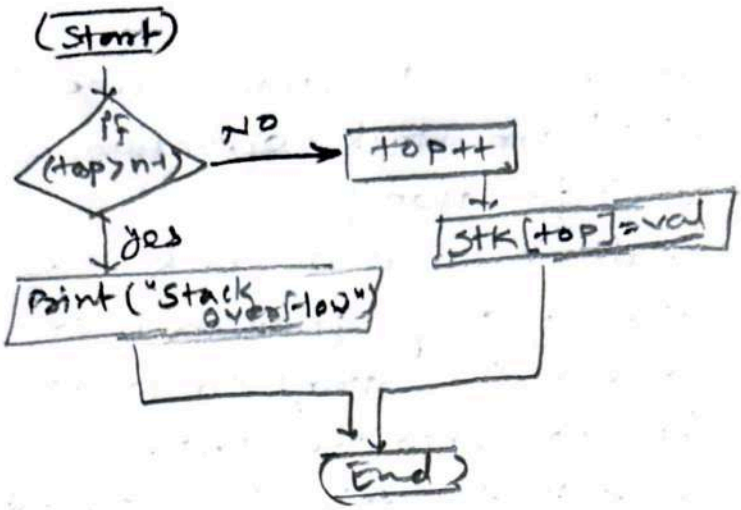
(e) Reverse & print the reversed string.

(10) End.

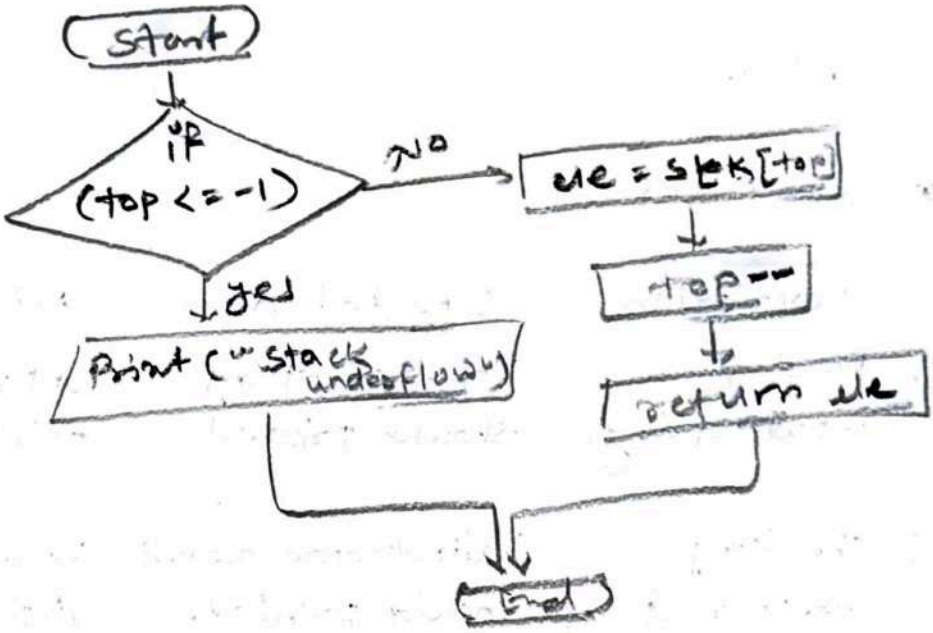
* Test Cases :-

	Description	Expected o/p	Actual o/p	Result
1)	Print the input string.	Inputed string should printed	Input string is printed.	Pass
2)	Checking whether string is palindrome or not.	Palindrome result should printed.	Result get displayed.	Pass
3)	Input : AOA	Palindrome : Yes	Palindrome : Yes	Pass
4)	Reverse string is printing or not.	Reverse string of inputed string should get displayed.	Reverse string get displayed.	Pass
5)	Input : ABC	Palindrome : NO	Palidrome : NO	Pass.

for push() -



for pop() -



DYP
Dr. D.Y. PATIL EDUCATIONAL COMPLEX
AKURDI, PUNE

* Conclusion :-

In this practical, we have successfully implemented palindrome string using stack along with various operations on stack.

* Title :- Write C++ program to check whether given expression is well parenthesized or not.

* Objective :- To check whether given expression is well parenthesized or not.

* Problem Statement :- In any programming languages, mostly syntax error occurs due to unbalancing delimiter such as (), {}, []. Write C++ program using stack to check whether the given expression is well parenthesized or not.

* Hardware & Software Requirements :-
1) Text editor
2) GCC / G++ compiler
3) Linux OS.
4) RAM - 8GB+
5) Processor - i3+ Gen
6) Disk space - 256GB+

* Theory :-
• Stack -

Stack is a abstract data type & linear data structure. It is data structure in which addition of new element or deletion of exiting element always takes place at a same end. This end is known as the top of the stack.

• Basic Operations of stack :-

(1) push()
(2) pop()

- Applications of stack :-
  1) Expression evolution
  2) Expression conversion
  3) Parsing
  4) Simulation of Recursion
  5) function call
  6) etc.

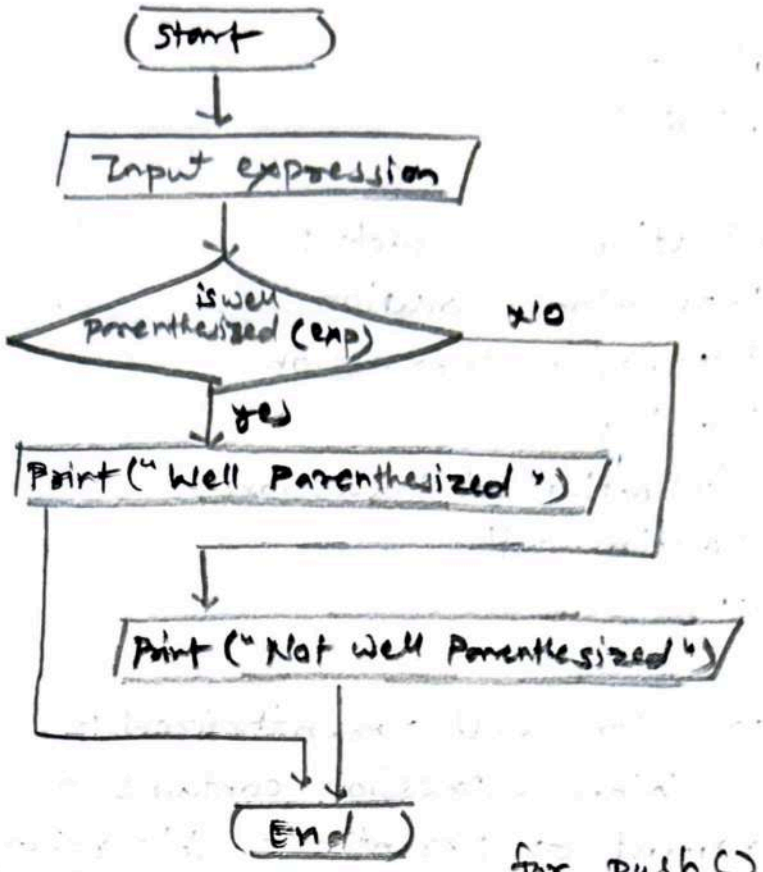
- Expression well parenthesized :-

When expression contains a balanced & proper arrangment of parenthesis. This means that for every opening parenthesis there is a corresponding closing parenthesis should be present.

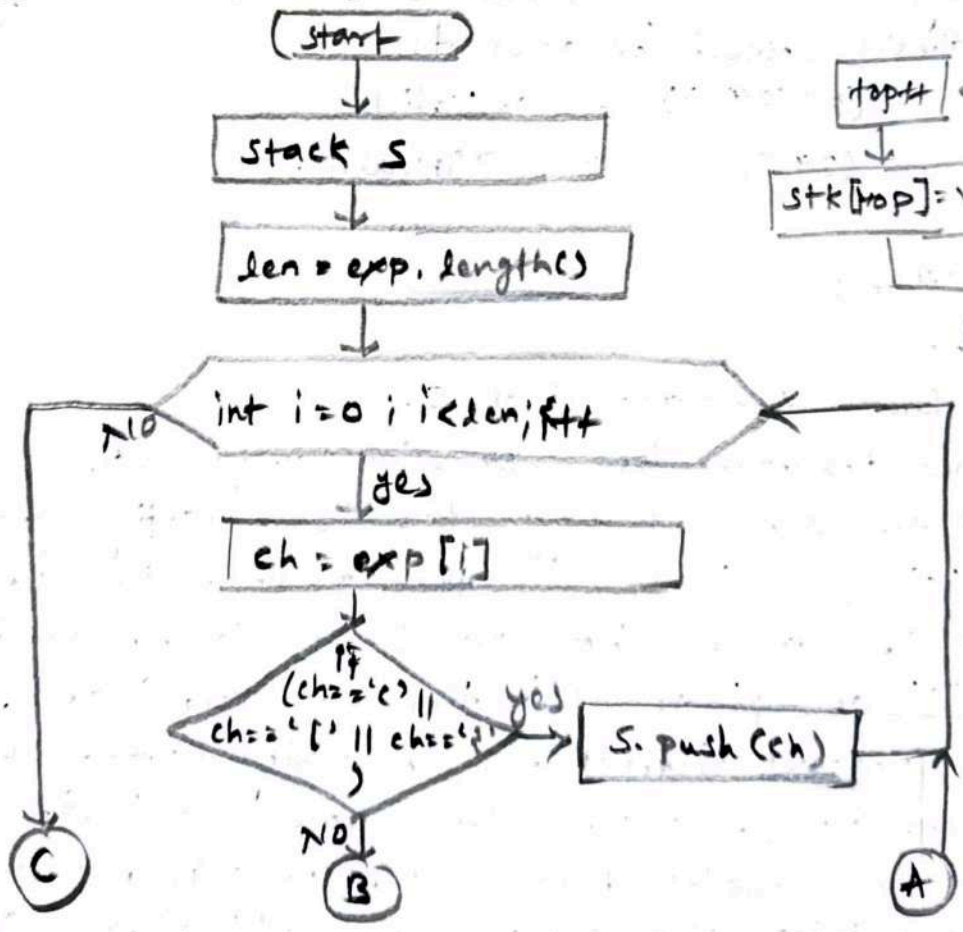
Ex. (2+(3*4)) ⟶ Valid
(2*5(⟶ Invalid.

* Algorithm :-
(1) Start
(2) Declare character stack S.
(3) Now, traverse the expression string exp.
(4) If the current character is a starting bracket ('(' or '{' or '[') then push it to stack.
(5) If the current character is a closing bracket (')' or '}' or ']') then pop from stack.
If popped character is matching starting bracket then fine, else parenthesis are not balanced.
(6) After complete traversal, stack is not empty then, Print "Not well parenthesized" otherwise "well parenthesized"

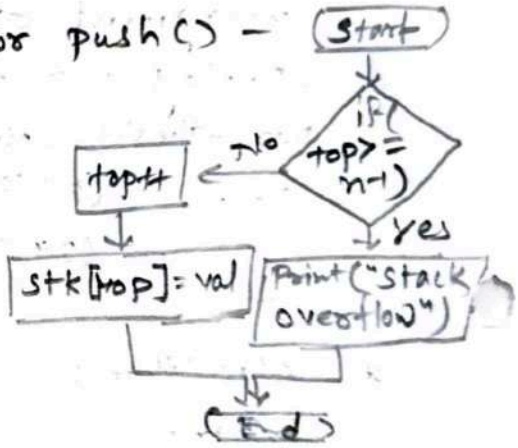
* Flowchart :- for main() :-



for isParenthesized() :-



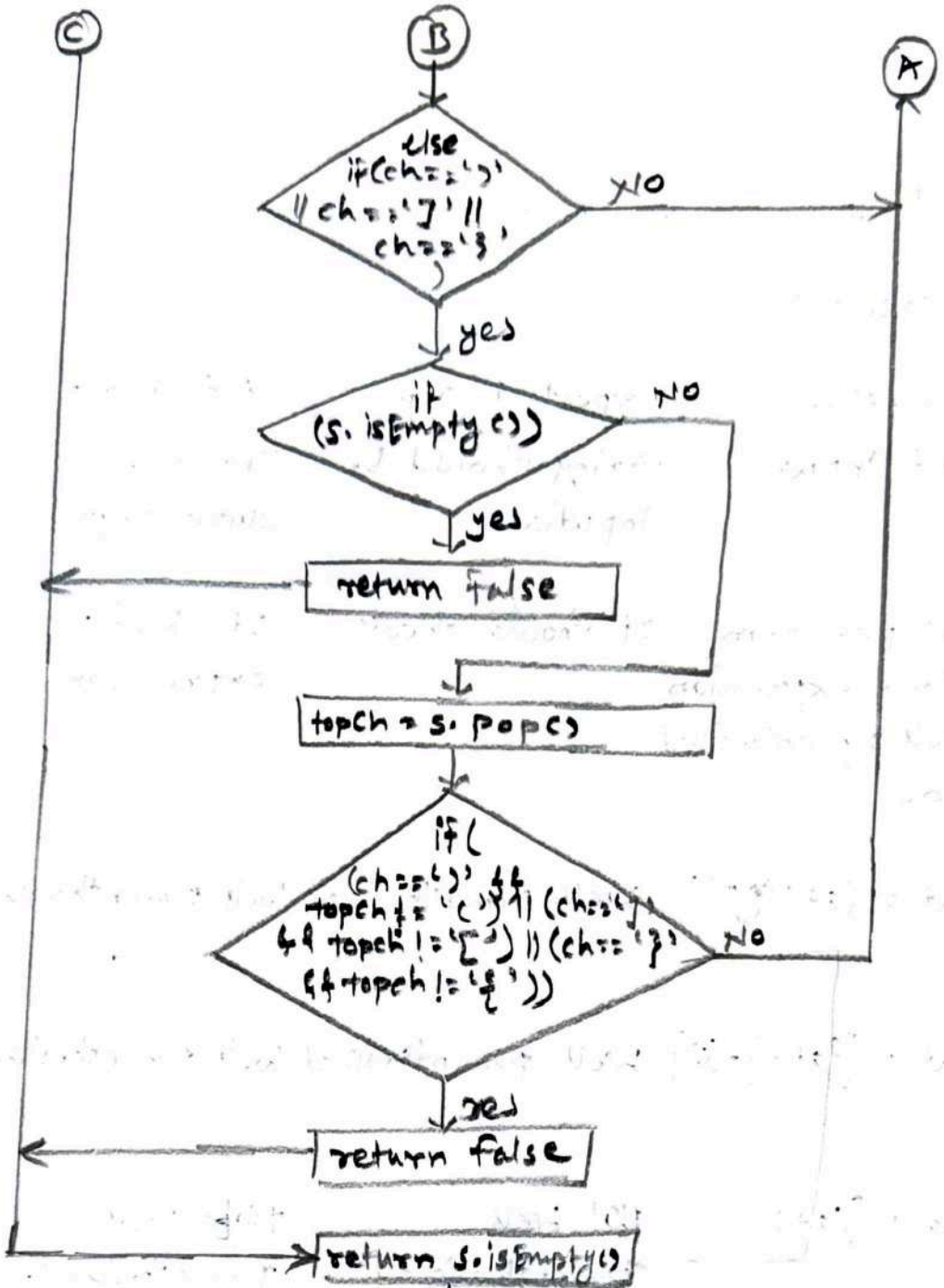
for push() -



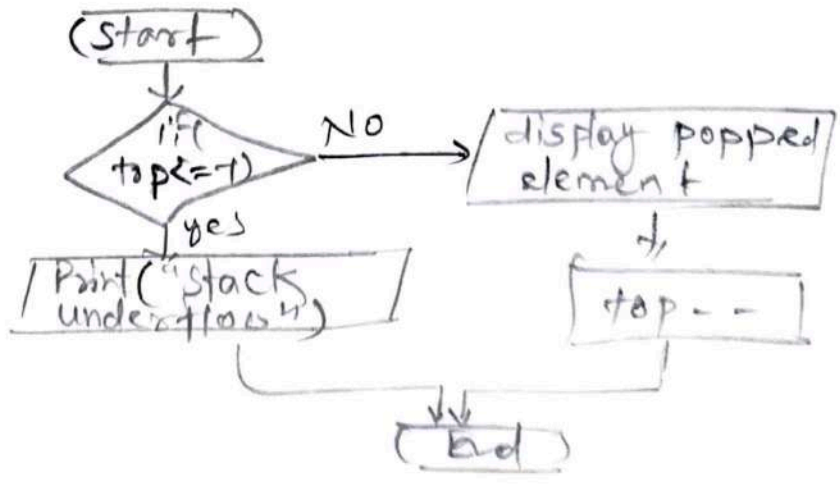
(7) End.

* Test Cases :-

	Description	Expected O/P	Actual O/P	Result
1)	Accept Input	String should be inputed.	Inputed successfully	Pass
2)	Check perform whether expression is well parenthesized or not.	It should check	It checks expression.	Pass
3)	Input : (a+b(*5))	Well parenthesized	Well parenthesized	Pass
4)	Input : [{a+b}-(c)]	Well parenthesized	Well parenthesized	Pass
5)	Input : [(a+b)	Not well parenthesized.	Not well parenthesized.	Pass


C
B
A
else if(ch=='>' || ch=='}' || ch=='}',
NO
yes
if (s.isEmpty())
NO
yes
return false
topch = s.pop()
if ((ch==')' && topch != '(') || (ch=='}' && topch != '[') || (ch=='}' && topch != '{'))
NO
yes
return false
return s.isEmpty()
End

for pop() -


Start
if(top<=-1)
NO
yes
Print("stack underflow")
display popped element
top--
End

DYP
Dr. D.Y. PATIL EDUCATIONAL COMPLEX
AKURDI, PUNE

* Conclusion :-

In this practical, the program is developed for the checking of expression that it is well parenthesized or not, successfully.

* Title :- Write C++ program for simulating job queue.

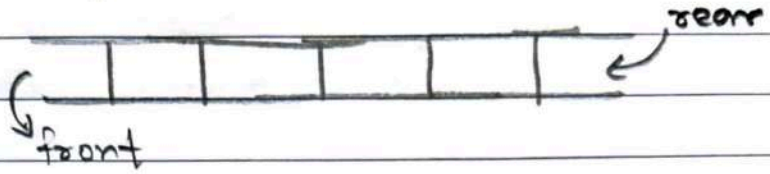
* Objectives :- 1) To study queue as data structure.
2) To understand implementation of simple queue, Priority queue & perform various operations on it.

* Problem Statement :- Queues are frequently used in computer programming, and a typical example is the creation of a job queue by an OS. If the OS does use priorities, then the jobs are processed in the order they enter the system. Write C++ program for simulating job queue. Write functions to add job & delete job from queue.

* Hardware & Software Requirements :-
1) C++ compiler.
2) Text editor
3) 64/32 bit Linux OS
4) RAM - 4GB+
5) ROM - 256 GB+

* Theory :-
• Queue :-
Queue is a linear data structure in which the insertion & deletion operations are performed at two different ends. In Queue, adding & removing of elements are performed at 2 different positions. The insertion is performed at one end & deletion is performed at other end. In Queue, the insertion operation is perform

at a position which is known as 'rear'. and the deletion operation is performed at position which is known as 'front.' It works on First In First Out (FIFO) principle.



- Basic operations on Queue :-

(1) In Queue, the insertion operation is performed using a function called as "enQueue()".

(2) In Queue, the deletion operation is performed using a function called as "deQueue()".

(3) In Queue, to display the elements "display()" function is used.

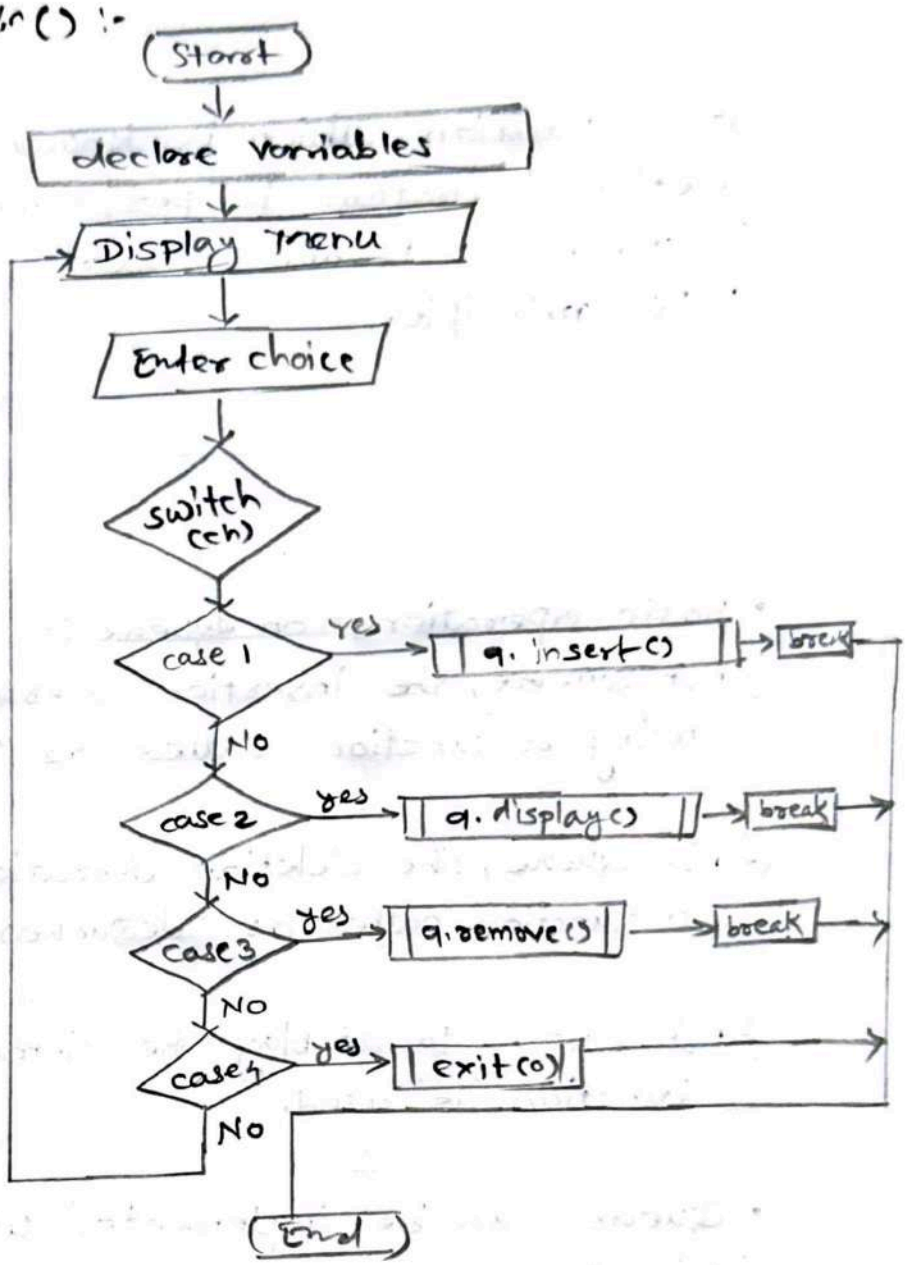
- Queue can be implemented using ;

(1) Array
(2) Linked List.

* Algorithm :-

(1) Start
(2) Create a structure 'job' to store job information, including job no. ('jno'), execution time ('t1'), start time('t2') & completion time ('t3').
(3) Initialize a queue to manage the jobs.
(4) Implement an 'insert()' function to add jobs to the queue by providing job no. & execution time

* Flowchart :- for main() :-

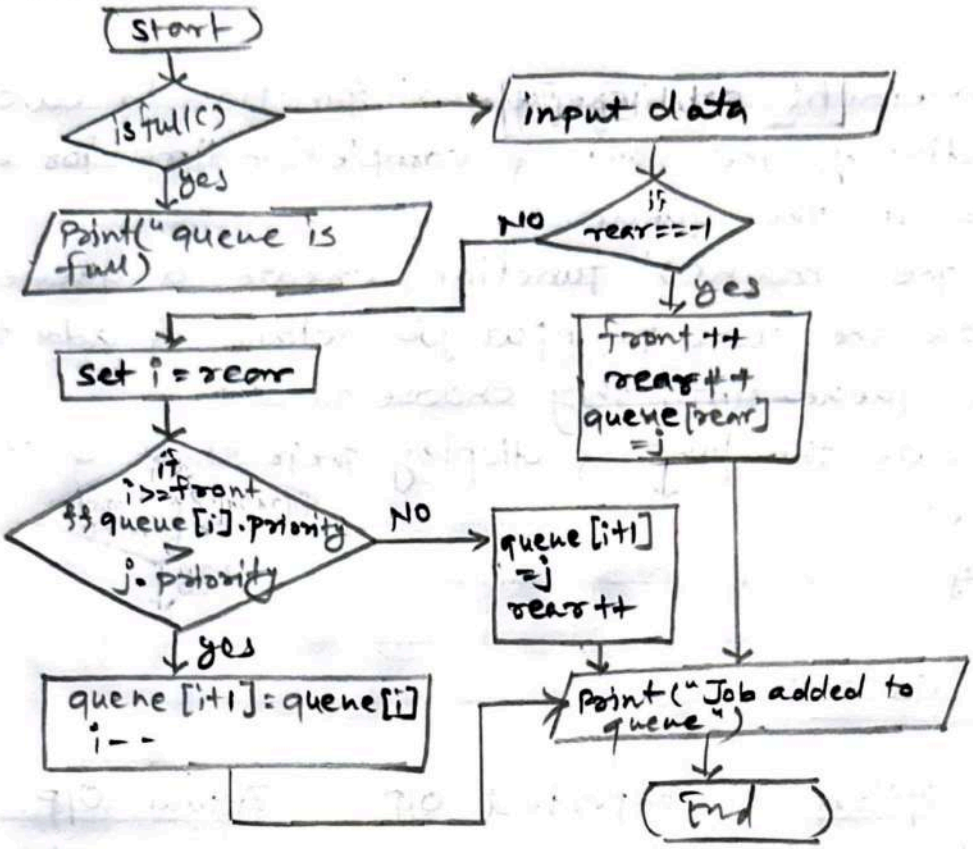


(5) Implement an 'execute()' function to calculate & display the start & completion time for each job in the queue.
(6) In the 'main()' function, create a queue object.
(7) Allow the user to input job details & add them to the queue until they choose to stop.
(8) Execute the jobs & display their start & completion times.
(9) Stop.

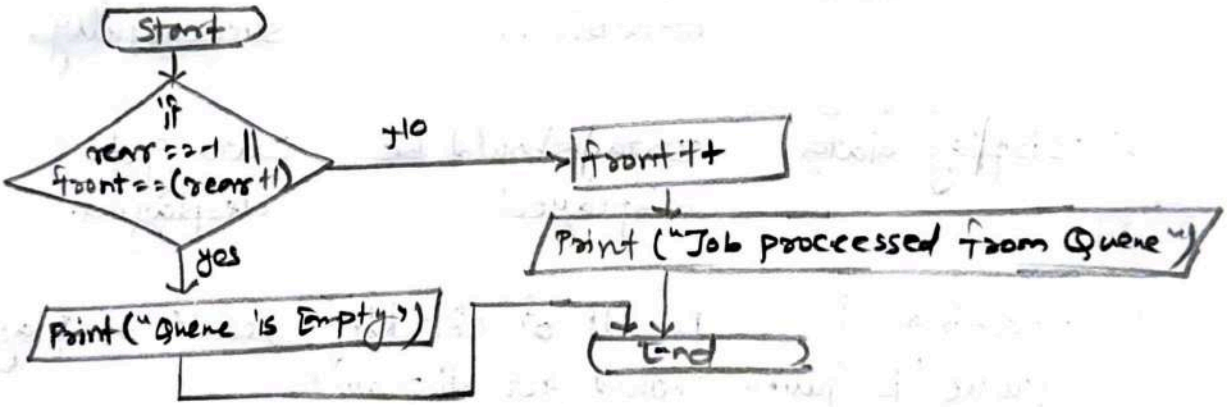
* Test Cases :-

	Description	Expected O/P	Actual O/P	Result
1)	Accept data	Data should be accepted.	Data inputed successfully.	Pass.
2)	Display data	Data should be displayed	Data get displayed.	Pass
3)	Checking if Queue is fulled.	Result of checking should get displayed.	Result displayed.	Pass
4)	Checking if Queue is empty.	Result of checking should get displaged.	Result displayed.	Pass
5)	Data deletion from queue.	Deletion result should get print.	Deletion result printed successfully	Pass.

for insert() :-



for remove() :-



DYP
Dr. D Y PATIL EDUCATIONAL COMPLEX
AKURDI, PUNE

* Conclusion :-

Hence, we have leaned how to implement Queue & perform various operations on it.

8/11/23

* Title :- To implement double-ended queue (dequeue)

* Objectives :- 1) To study double-ended queue.
2) To understand implementation of double-ended queue. & perform various operations on it.

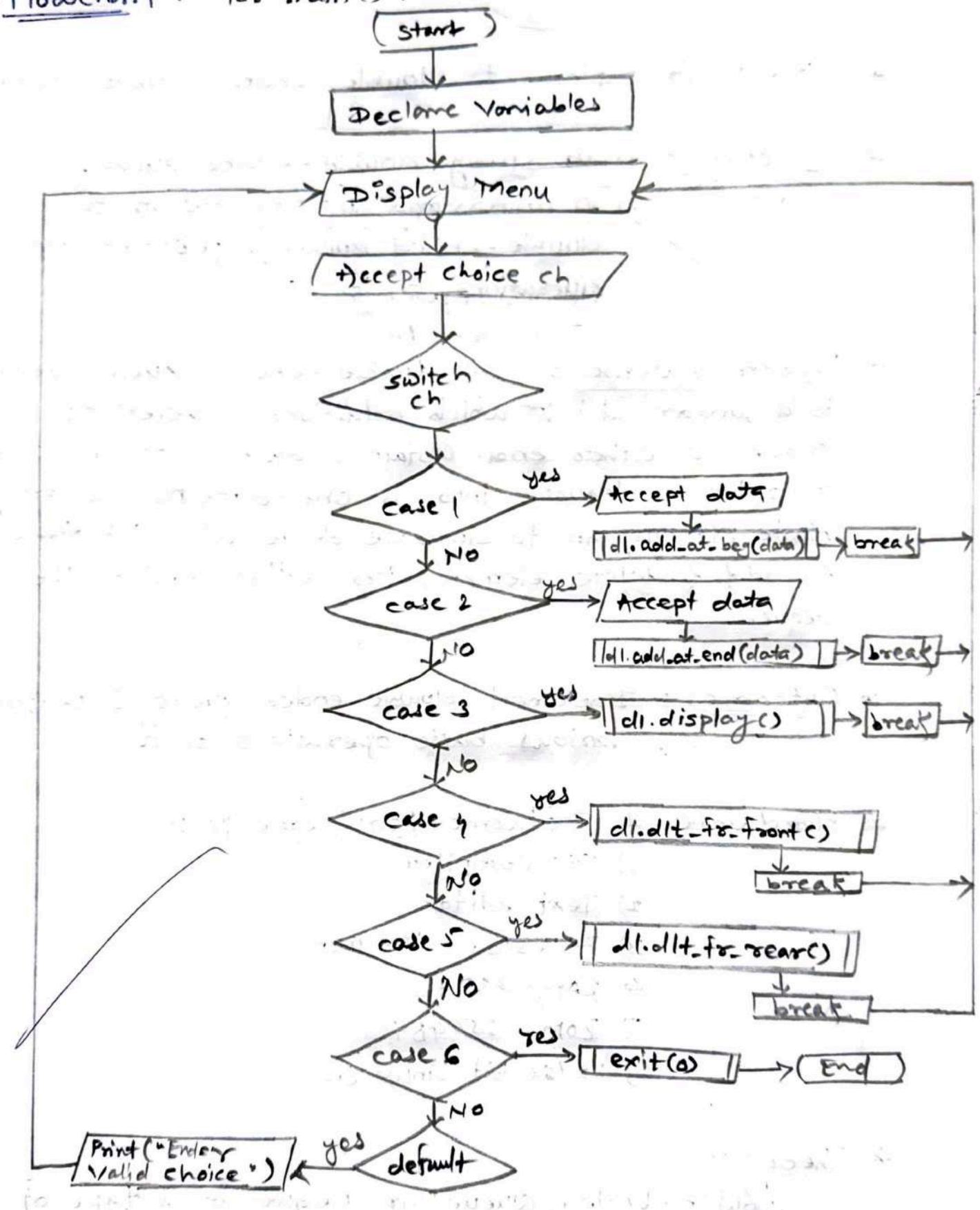
* Problem Statement :- A double-ended queue (dequeue) is a linear list in which additions & deletions may made at either end. Obtain a data representation mapping a dequeue into a one-dimensional array. Write C++ program to simulate deque with functions to add & delete elements from either end of the deque.

* Outcomes :- Implement double-ended queue & perform various basic operations on it.

* Hardware & Software Requirements :-
1) C++ Compiler
2) Text editor
3) Processor - i3+ Gen
4) RAM - 4GB+
5) ROM - 256GB+
6) 32/64 bit Linux Os

* Theory :-
Double-Ended Queue or Deque is a type of queue in which insertion & removal of elements can be performed from either from the front / rear.

* Flowchart :- for main() :-



Thus, it doesn't follow FIFO (First In First Out) rule.

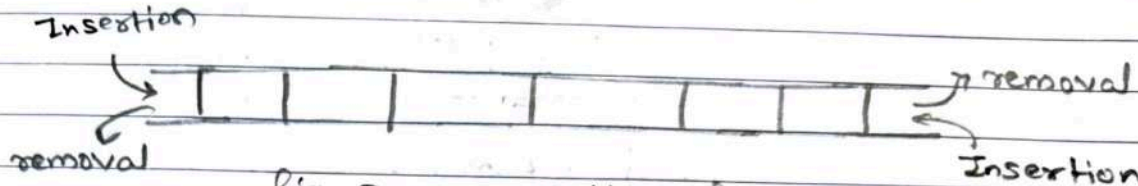


fig. Representation of Deque

- Types of Deque :-

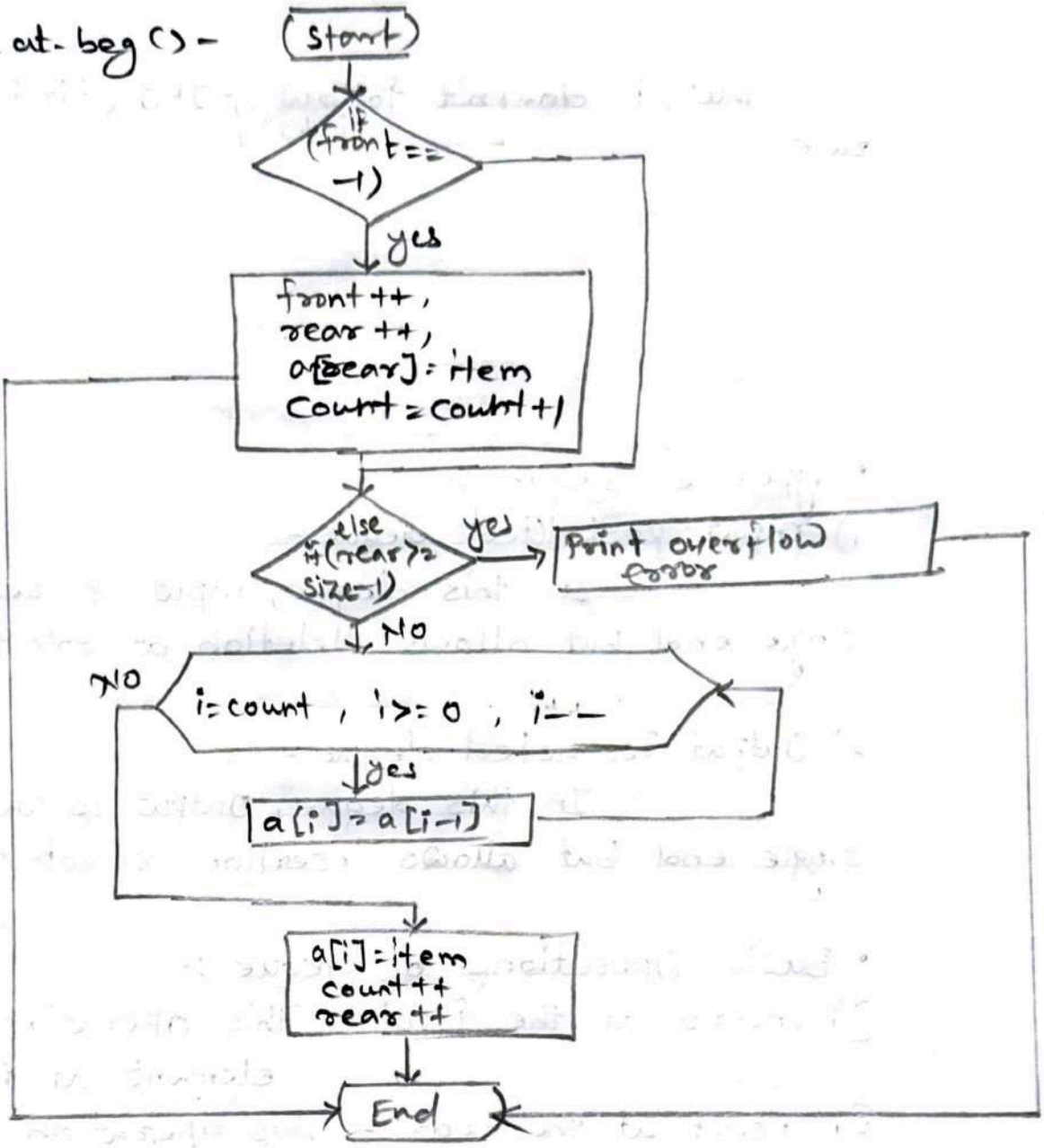
1) Input Restricted deque -
In this deque, input is restricted at a single end but allows deletion at both the ends.

2) Output Restricted deque -
In this deque, Output is restricted at a single end but allows insertion at both the ends.

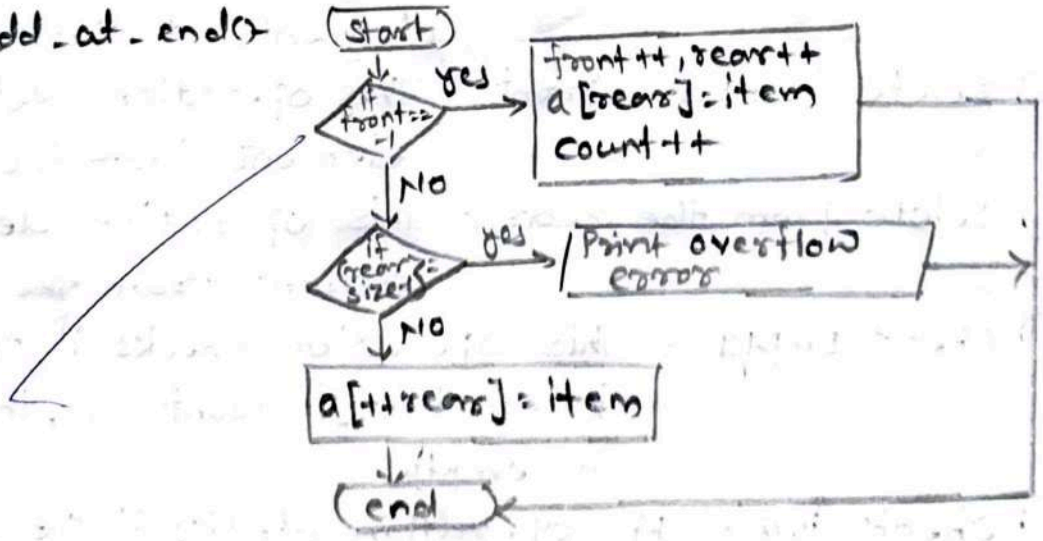
- Basic Operations of deque :-

(1) Insert at the front - This operation adds an element at the front.
(2) Insert at the rear - This operation adds an element at the rear.
(3) Delete from the front - This operation deletes an element from the front.
(4) Delete from the rear - This operation deletes an element from the rear.
(5) Check Empty - This operation checks if the deque is empty. If front = -1, the deque is empty.
(6) Check full - This operation checks if the deque is full. If front = 0 & rear = n-1 or front = rear+1, the deque is full.

for add_at_beg() -



for add_at_end() -

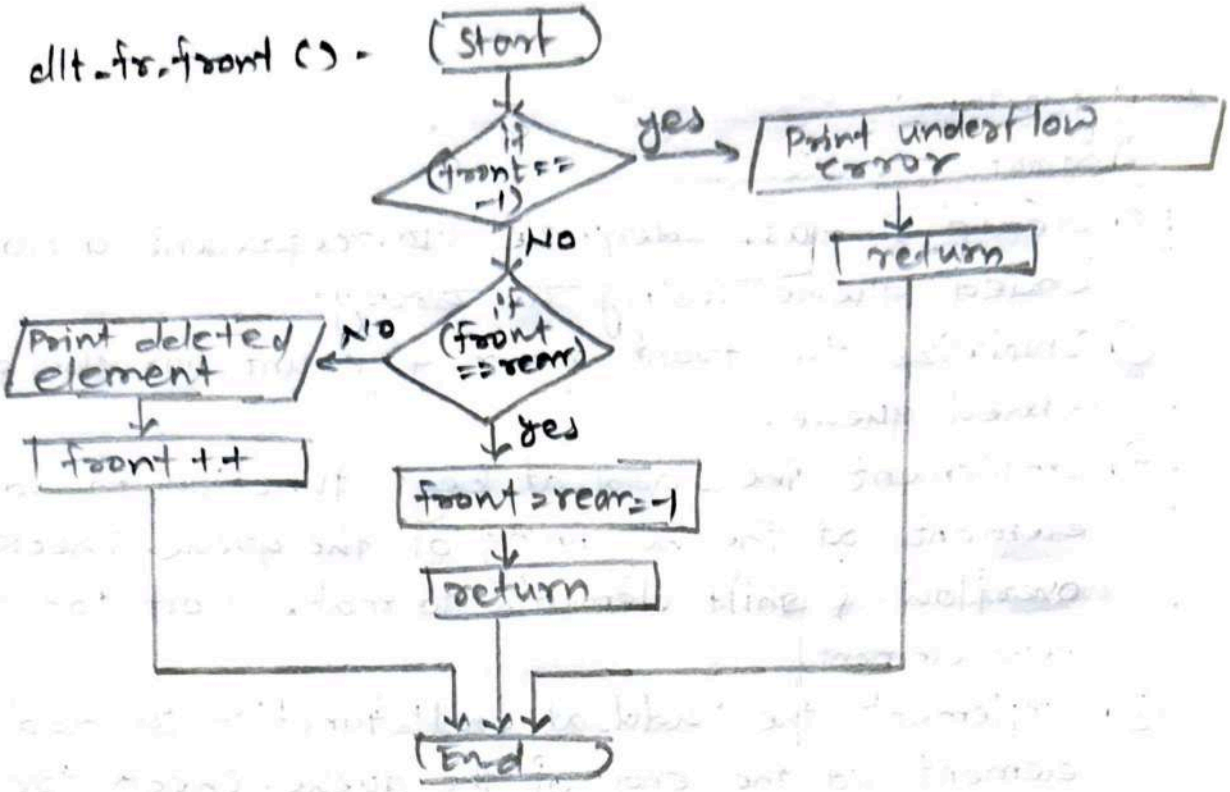


DYP
Dr. D Y PATIL EDUCATIONAL COMPLEX AKURDI, PUNE

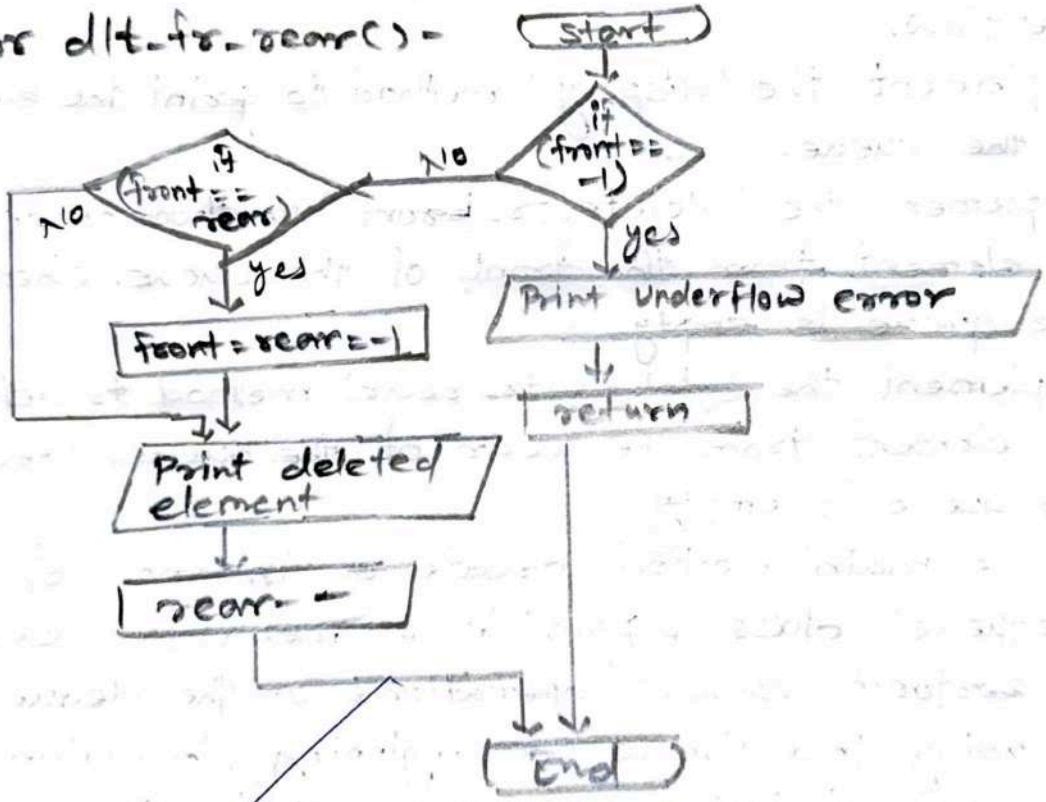
* Algorithm :-

(1) Start
(2) Create a class 'dequeue' to represent a double-ended queue using an array.
(3) Initialize the front, rear & count for the double-ended queue.
(4) Implement the 'add_at_beg' function to add an element at the beginning of the queue. Check for overflow & shift elements to make room for the new element.
(5) Implement the 'add_at_end' function to add an element at the end of the queue. Checks for overflow.
(6) Implement the 'display' method to print the elements in the queue.
(7) Implement the 'delete_fr_front' method to delete an element from the front of the queue. Check if the queue is empty.
(8) Implement the 'delete_fr_rear' method to delete an element from the rear of the queue. check if the queue is empty.
(9) In the 'main' method, create an instance of the 'dequeue' class & provide a menu for users to perform various operations on the deque, including insertion at the beginning, insertion at the end, displaying the queue, deletion from the front, deletion from the rear, and exiting.
(10) Use loop to continuously prompt the user for their choice & perform the selected operation until they choose to exit.
(11) Stop.

for dlt_fr_front() -



for dlt_fr_rear() -



DYP
Dr. D Y PATIL EDUCATIONAL COMPLEX
AKURDI, PUNE

* Test Cases :-

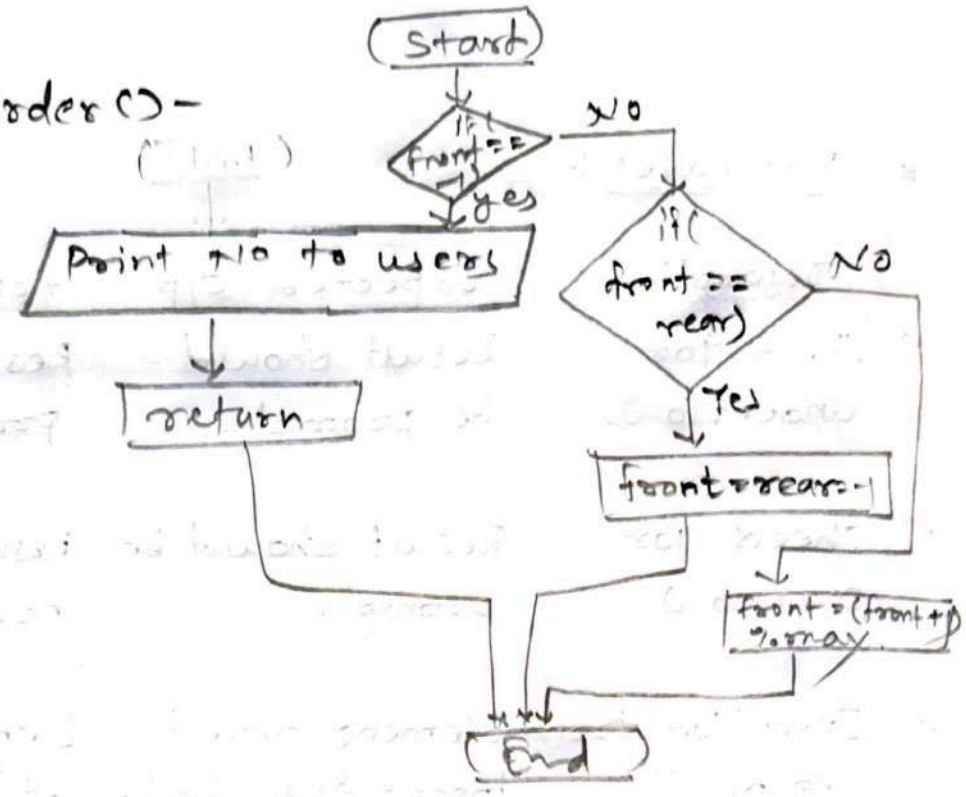
	Description	Expected O/p	Actual O/p	Result
1)	Check for underflow.	Result should be prompt.	Result get prompted	Pass
2)	Checks for overflow	Result should be prompt.	Result get prompted.	Pass
3)	Insertion from begin.	Element should insert from begin of dequeue.	Element inserted at begin.	Pass
4)	Insertion from end.	Element should insert from end of dequeue.	Element inserted at end.	Pass
5)	Deletion from front	front element should be deleted.	front element get deleted.	Pass
6)	Deletion from end	End element should be deleted.	End element get deleted.	Pass

* Conclusion :-

Hence, we have learned how to implement double-ended queue & perform various operations on it.

8/11/23

for serveOrder()-



DYP

Dr. D.Y. PATIL EDUCATIONAL COMPLEX
AKURDI, PUNE

(4)

* Title :- To implement circular queue.

* Objectives :- 1) To study circular queue
2) To understand implementation of circular queue & perform various operations on it.

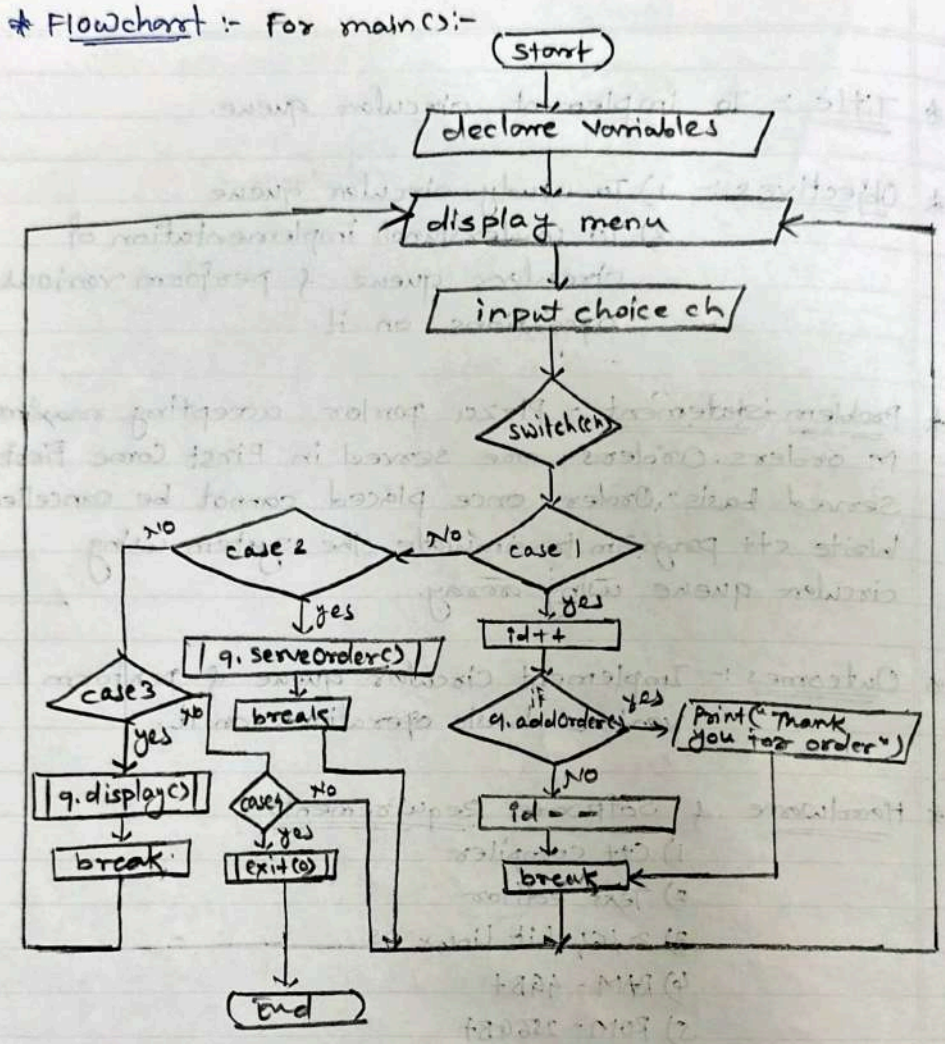
* Problem statement :- Pizza parlor accepting maximum M orders. Orders are served in First Come First Served basis. Order once placed cannot be cancelled. Write c++ program to simulate the system using circular queue using array.

* Outcomes :- Implement circular queue & perform various basic operations on it.

* Hardware & Software Requirements :-
1) C++ compiler
2) Text editor
3) 32/64 bit Linux OS
4) RAM - 4GB+
5) ROM - 256GB+
6) Processor - 1st GEN

* Theory :-
In a normal queue, we can insert elements until queue becomes full. But, once if queue becomes full, we can't insert the next element until all the elements are deleted from the queue.

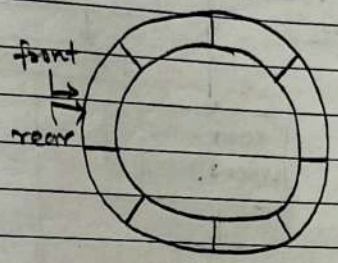
* Flowchart :- For main():-



DYP

• Circular Queue :-

It is a type queue can be defined as a linear datastructure in which the operations are performed based on FIFO (First In First Out) principle & the last position is connected back to the first position to make a circle.



• Basic operations on Circular Queue :-

(1) enQueue() - It is used to insert an element into the circular queue.

(2) deQueue() - It is used to delete an element from the circular queue.

(3) display() - It is used to display elements of circular queue.

* Algorithm :-

(1) Start

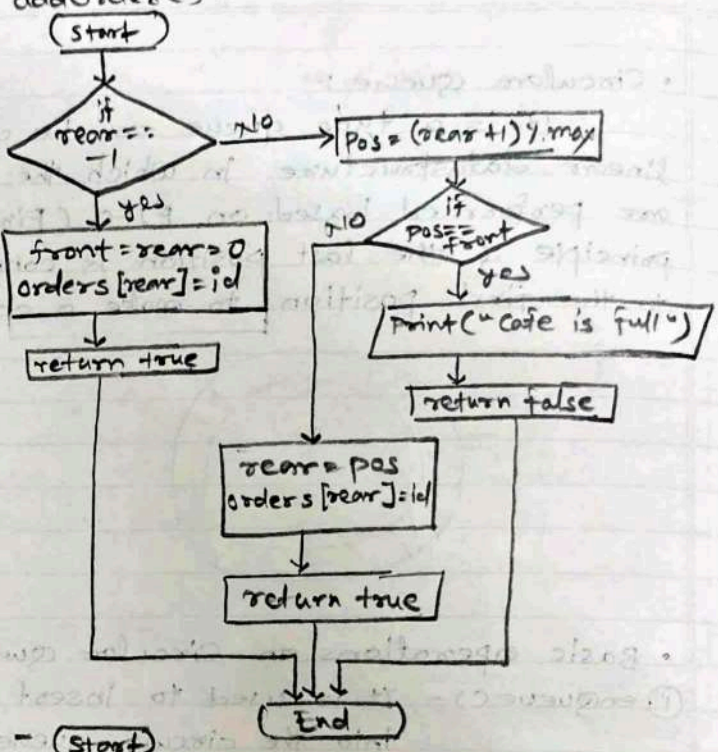
(2) Create a structure 'piz' to represent pizza orders with order no. 'ono' & type of pizza 'ot'

(3) Create a 'queue' class with an array of 'piz' objects, along with counters & indices to manage the queue.

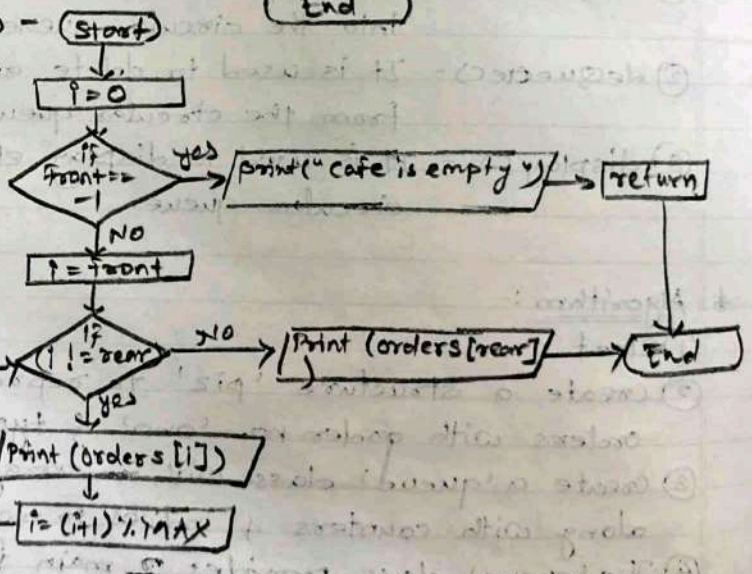
(4) The 'queue' class provides 2 main function :-

(a) insert() - Allows user to place pizza orders. It checks for overflow & adds orders to the queue.

for addOrder() -



for display() -



(b) display() Allows the user to display pizza orders. It prompts for the no. of orders to display & shows the order details.

(c) ~~display() Allows the user~~

(5) In the 'main()' function, creates an instance of the 'queue' class & enter a loop to interact with the user.

(6) End.

* Test Cases :-

	Description	Expected O/P	Actual O/P	Result
1)	Accept data	Data should get accepted.	Data accepted successfully.	Pass
2)	Display data	Displaying of data should be done.	Data displayed successfully.	Pass.
3)	Accept order.	Order should be accepted except overflow.	Order got accepted.	Pass
4)	Exit program	Thank you for using our software	message get displayed.	Pass

DYP
Dr. D Y PATIL EDUCATIONAL COMPLEX
AKURDI, PUNE

* Conclusion :-

Hence, we have learned how to implement circular queue & perform various operations on it.

9/11/23