A Laboratory Manual

for

Data Structures Using 'C'

(22317)

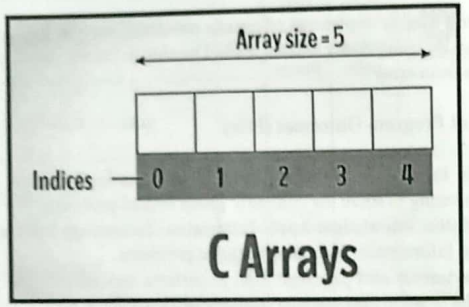
Semester-III

(CO/CM/CW/IF)

Maharashtra State
Board of Technical Education, Mumbai
(Autonomous) (ISO:9001:2015) (ISO/IEC 27001:2013)

VII. Minimum Theoretical Background

Array:



An array is a collection of data that holds fixed number of values of same type. For example: if you want to store marks of 5 students, you can create an array for it.

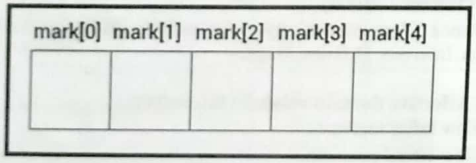
Declaration of an array in C:

Syntax: data_type array_name [array_size];

Ex. : int mark[5];

Array elements are accessed by indices.

Suppose an array *mark [5] declared* as above. The first element is *mark [0]*, second element is *mark [1]* and so on.



- Arrays have 0 as the first index not 1. In this example, *mark[0]*
- If the size of an array is *n*, then to access the last element, (n-1) index is used. In this example, *mark[4]*
- Suppose the starting address of mark [0] is 2120. Then, the next address, mark [1], will be 2122, address of mark [2] will be 2124 and so on. It's because the size of a int is 2 bytes.

VIII. Algorithm:

Algorith to insert the new element at specific index in one dimension array is given below:

step 1: Read the index position & value of element to be inserted.

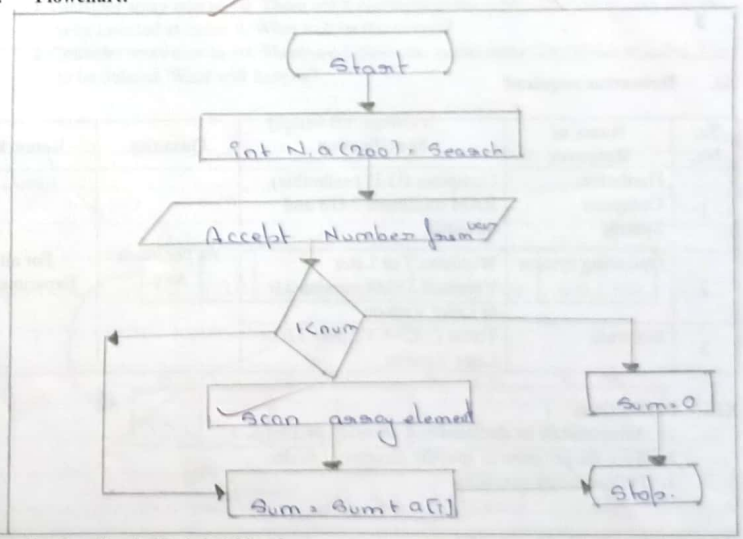
step 2: If Current no of element are less than maximum size of array that insert operation can be performed.
if (N < (max-1)

step 3: Shift / Move one element one position head N to index.

step 4: Update Current no of element N = N + 1 End of if.

step 5: Finish.

IX. Flowchart:



X. C Program Code:

```
#include <stdio.h>
#include <conio.h>
main()
{
    int a[20], n;
    clrscr();
    for(i=0; i<=19; i++)
    {
        printf("Enter array elements");
        scanf("%d", a[i]);
    }
    for(i=0; i<=19; i++)
    {
        printf("Array element %d", a[i]);
    }
getch();
return 0;
}
```

XI. Resources required

Sr. No.	Name of Resource	Specification	Quantity	Remarks
1	Hardware: Computer System	Computer (i3-i5 preferable), RAM minimum 2 GB and onwards	As per batch size	For all Experiments
2	Operating system	Windows 7 or Later Version/LINUX version 5.0 or Later Version		
3	Software	Turbo C /C++ Version 3.0 or Later Version		

XII. Precautions

1. Array should be declared and accessed properly.
2. Save the program in specific directory / folder.
3. Follow safety practices.

XIII. Resources used

S. No.	Name of Resource	Specification
1	Computer System with broad specifications	Corie i3, 4 GB RAM
2	Software	Turbo C++
3	Any other resource used	Windows 7.

XIV. Results (Output of the Program)

a[0] = 1
a[1] = 22
a[2] = 33
a[3] = 44

XV. Conclusion(s)

0	1	2	3	4	5	6	7	8	9
10	15	20	23	86	94	101	12	16	18

XVI. Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

1. Consider array size as 10. There are 8 elements in the array. The value (say 100) is to be inserted at index 9. What will be the output?
2. Consider array size as 10. There are 6 elements in the array. The value at index 7 is to be deleted. What will happen?

(Space for answers)

1) Ans:

0	1	2	3	4	5	6	7	8
10	20	30	40	50	60	70	80	90

0	1	2	3	4	5	6	7	8
10	20	30	40	50	60	80	90	

value will be deleted

XVII. Exercise

1. Perform the given operations on the following array and show diagrammatic presentation of each operation given below:

0	1	2	3	4	5	6	7	8	9	10
10	20	30	40	50	60	70	80	90		

   a) Delete element at index 8
   b) Add element at index 4 with value 999
   c) Delete element at index 0

2. Insert Array2 at index 2 of Array1 :

Array1:

0	1	2	3	4	5	6	7	8	9	10
10	20	30	40	50						

Array2:

0	1	2
100	200	300

(Space for answers)

1) Ans:

a)

0	1	2	3	4	5	6	7	8	9
10	20	30	40	50	60	70	80	90	100

Delete the value of in.

b)

0	1	2	3	4	5	6	7	8	9
10	20	30	40	50	60	70	80		

Insert the value in it.

c)

0	1	2	3	4	5	6	7	8	9	10
	20	30	40	50	60	70	80	90		

Delete the value of index

XVIII. References / Suggestions for further Reading

1. https://www.tutorialspoint.com/cprogramming/c_arrays.htm (as on 17/01/2018)
2. https://youtube (as on 17/01/2018)

XIX. Assessment Scheme

Performance indicators		Weightage
Process related(10 Marks)		**30%**
1.	Debugging ability	20%
2.	Follow ethical practices.	10%
Product related (15 Marks)		**70%**
3.	Correctness of algorithm	15%
4.	Correctness of Program codes	25%
5.	Quality of input/output messaging and output formatting	20%
6.	Timely Submission of report	5%
7.	Answer to sample questions	5%
Total (25 Marks)		**100%**

List of Students /Team Members

1. .....
2. .....
3. .....
4. .....

Marks Obtained			Dated signature of Teacher
Process Related(10)	Product Related(15)	Total(25)	
8	11	19	

Practical No. 2: Search a Data Using Linear Search

I. Practical Significance

In order to perform some operation on specific data element, the data has to be searched in data collection, and the system requires following a searching method. One of the most commonly used method is linear search for searching data from the given list.

II. Relevant Program Outcomes (POs)

- **Basic knowledge:** Apply knowledge of basic mathematics, sciences and basic engineering to solve the broad-based Computer engineering problem.
- **Discipline knowledge:** Apply Information Technology knowledge to solve broad-based Information Technology related problems.
- **Experiments and practice:** Plan to perform experiments, practices and to use the results to solve Information Technology related problems.
- **Engineering tools:** Apply appropriate Computer Engineering / Information Technology related techniques/ tools with an understanding of the limitations.
- **Communication:** Communicate effectively in oral and written form.

III. Competency and Practical skills

This practical is expect to develop the following skills in you

Develop 'C' programs to solve broad-based computer group related problems.

1. Write algorithm and draw Flow Chart for Linear Search.
2. Write/Compile/Debug and Save C program for function to search using Linear Search.

IV. Relevant Course Outcome(s)

- Apply different searching and sorting techniques.

V. Practical Outcome (PrOs)

Implement a 'C' program to search a particular data from the given Array using Linear Search.

VI. Relevant Affective domain related Outcome(s)

1. Follow safety measures
2. Follow ethical practices.

VII. Minimum Theoretical Background

Searching: Searching is just finding out the data item among given list. Two cases are possible that either item is found or item not present in the list.

Linear search: Linear Search is a very simple search algorithm. In this sequential search is done by searching items one by one. Each item is checked and if a match is found then that particular item is returned, otherwise the search is continues till end of

XI. Resources required

Sr. No.	Name of Resource	Specification	Quantity	Remarks
1	Hardware: Computer System	Computer (i3-i5 preferable), RAM minimum 2 GB onwards	As per batch size	For all Experiments
2	Operating system	Windows 7 or Later Version/LINUX version 5.0 or Later Version		
3	Software	Turbo C /C++ Version 3.0 or Later Version		

XII. Precautions

1. Save the program in specific directory / folder.
2. Follow safety practices.

XIII Resources used

S. No.	Name of Resource	Specification
1	Computer System with broad specifications	Core i3, 4GB RAM.
2	Software	Turbo C++
3	Any other resource used	Windows 7

XIV Result (Output of the Program)

A[0] = 1
A[1] = 3
A[2] = 5
A[3] = 7
A[4] = 9

XV Conclusion(s)

XVI Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

1. Linear Search is most suitable for which kind of data list?
2. What is the output if list of item having same data item which is to be searched?

(Space for answers)

a) 1. Read list of N numbers
2. Print list of N numbers
3. Linear Search method.

b)

0	1	2	3	4	5	6	7	8	9
89	11	12	81	66	42	31	38	64	90

Starting with index = 0 the current element is compared with with key 66 is found at index position 5.

XVII Exercise

1. Consider following list to perform Linear Search
   56, 36, 89, 56, 01,00,67,59
   i. Search the item 01 from above list and write the item is found or not with procedure.
   ii. Search the item 55 from above list. Write the item is found or not with procedure.
2. State the limitations of Linear Search in terms of Time Complexity.

(Space for answers)

1) Ans:)

0	1	2	3	4	5	6	7	8
56	36	86	92	91	87	54	90	8

2) Ans:)

0	1	2	3	4	5	6	7	8
56	36	86	92	91	87	54	90	8

The key 36 is found is index 1.

3) Ans:)

0	1	2	3	4	5	6	7	8
56	36	86	92	91	87	54	90	8

The key is not found at index 5

XVIII References / Suggestions for further Reading

1. https://www.tutorialspoint.com/data_structures_algorithms/linear_search_algorithm.htm (as on 17/1/2018)
2. https://www.youtube.com/watch?v=hi-lwJRQ1-s
3. https://www.youtube.com/watch?v=iwo5WAldDks

XIX Assessment Scheme

	Performance indicators	Weightage
	Process related(10 Marks)	**30%**
1	Debugging ability	20%
2	Follow ethical practices.	10%
	Product related (15 Marks)	**70%**
3	Correctness of algorithm	15%
4	Correctness of Program codes	25%
5	Quality of input/output messaging and output formatting	20%
6	Timely Submission of report	5%
7	Answer to sample questions	5%
	Total (25 Marks)	**100%**

List of Students /Team Members

1. .....
2. .....
3. .....
4. .....

Marks Obtained			Dated signature of Teacher
Process Related(10)	Product Related(15)	Total(25)	
8	11	19	

Fig 3. Upper portion of the array

We change our low to mid+1 and find the new mid value again

Low=mid+1

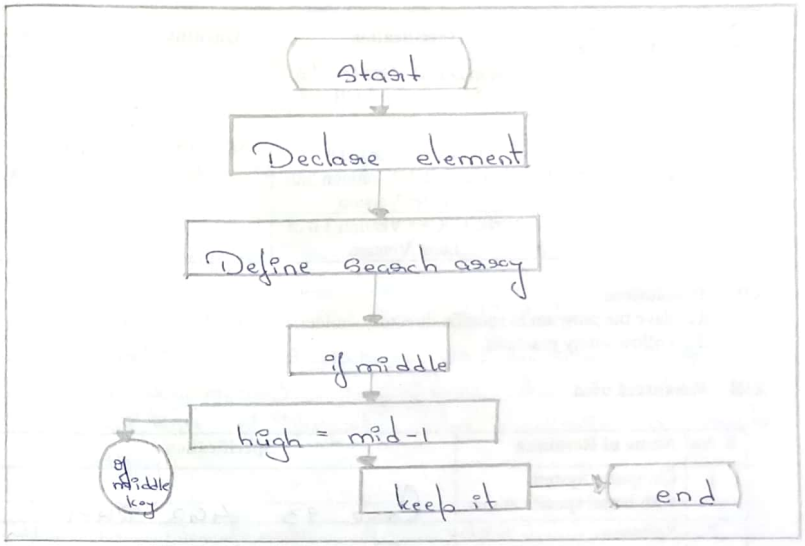
Mid=low + (high-low) /2

Do above given procedure till the item is found if present in the list.

VIII. Algorithm

1. Start.
2. Declare array element.
3. Define key element search key.
4. Find middle element
5. if (key < middle)
   set ~~hug~~ high = middle - 1;
   else {
   set high = middle + 1;
6. Repeat
7. if (middle == key) then
   write " Search is found.".

IX. Flow Chart



X. 'C' Program Code

```
#include <stdio.h>
#include <conio.h>
main()
{
   int first, last, middle, n, search, array[100];
   clrscr();
   printf("Enter number of element");
   scanf("%d", &n);
   for (i=0; i<n; i++)
   {
      for( j=0; j<n-1-1; j++
   { if (x[j] > x[j+1])
   { temp = x[j]
     x[j] = x[j+1]
   } } }  x[j+1] = temp;
   getch();
}
```

XI. Resources required

Sr. No.	Name of Resource	Specification	Quantity	Remarks
1	Hardware: Computer System	Computer (i3-i5 preferable), RAM minimum 2 GB and onwards	As per batch size	For all Experiments
2	Operating system	Windows 7 or Later Version/LINUX version 5.0 or Later Version		
3	Software	Turbo C /C++ Version 3.0 or Later Version		

XII Precautions

1. Save the program in specific directory / folder.
2. Follow safety practices.

XIII Resources used

S. No.	Name of Resource	Specification
1	Computer System with broad specifications	Core i3 4GB RAM
2	Software	Turbo C++
3	Any other resource used	Windows 7

XIV Result (Output of the Program)

10, 20, 30, 40, 50, 60

XV Conclusion(s)

Give a the sort array of element you can use an binary Linear search to located in element in the array

XVI Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

1. Binary Search is most suitable for what kind of data list.
2. If mid=4.5 then what should be the value of mid?

(Space for answers)

1) Read the list of N number in the assending order.

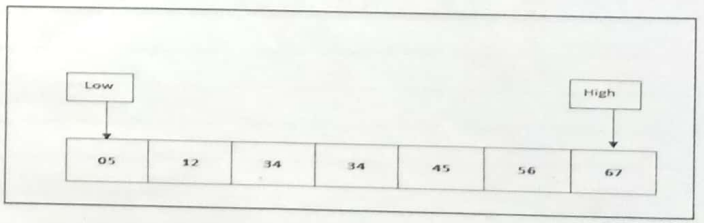
Input the list of N number Binary Search method

2) Binary Search is a half interval search, logarithmic search, or binary chop, is a search algorithm that finds the position of target value within a sorted array.

3) 12 is found at index 1.
69 is found at index 7.
100 is not found.

XVII Exercise

1. State the benefits and limitations of Binary Search in terms of Time Complexity.
2. Consider a following list and write down the steps to perform Binary Search to search following numbers.
   i. 12
   ii. 67
   iii. 100



(Space for answers)

If they are not equal, the half in which the target value cannot lie is eliminated and the search continues on the remaining half, again taking the middle element to compare to the target value. If the search end the target is not found.

XVIII References / Suggestions for further Reading

1. https://www.tutorialspoint.com/data_structures_algorithms/linear_search_algorithm.htm
2. https://www.youtube.com/watch?v=P3YID7liBug
3. https://www.youtube.com/watch?v=UeUyTbtFxQQ

XIX Assessment Scheme

Performance indicators		Weightage
Process related(10 Marks)		**30%**
1	Debugging ability	20%
2	Follow ethical practices.	10%
Product related (15 Marks)		**70%**
3	Correctness of algorithm	15%
4	Correctness of Program codes	25%
5	Quality of input/output messaging and output formatting	20%
6	Timely Submission of report	5%
7	Answer to sample questions	5%
Total (25 Marks)		**100%**

List of Students /Team Members

1. .....
2. .....
3. .....
4. .....

Marks Obtained			Dated signature of Teacher
Process Related(10)	Product Related(15)	Total(25)	
8	11	19	

Practical No. 4: Program to Sort an Array Using Bubble Sort

I. Practical Significance:

In order to perform some operation on specific data element, the data has to be sorted in data collection, and the system requires following a Sorting method. One of the most commonly used methods is Bubble Sort for sorting data from the given list.

II. Relevant Program Outcomes (POs)

- **Basic knowledge:** Apply knowledge of basic mathematics, sciences and basic engineering to solve the computer group related problems.
- **Discipline knowledge:** Apply Information Technology knowledge to solve broad-based Information Technology related problems.
- **Experiments and practice:** Plan to perform experiments and practices to use the results to solve the computer group related problems.
- **Engineering tools:** Apply relevant Computer programming / technologies and tools with an understanding of the limitations.
- **Communication:** Communicate effectively in oral and written form.

III. Competency and Practical skills

This practical expects to develop the following skills in the student.

Develop 'C' programs to solve computer group related problems.

1. Write algorithm and draw flow chart for Bubble sort.
2. Write / Compile / Debug and execute 'C' program for Bubble sort.

IV. Relevant Course Outcome(s)

- Apply different searching and sorting techniques.

V. Practical Outcome (PrOs)

- Implement a 'C' program to sort an array using Bubble Sort method.

VI. Relevant Affective domain related Outcome(s)

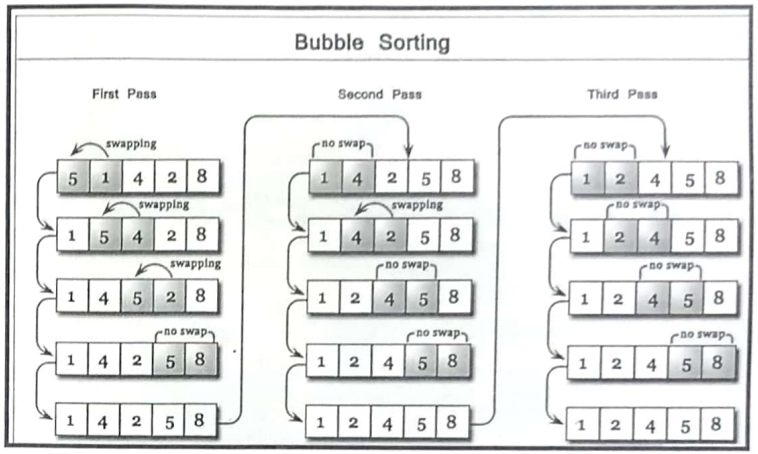
1. Follow safety measures.
2. Follow ethical practices.

VII. Minimum Theoretical Background

In Bubble Sort method the list is rearranged by exchanging the two adjacent elements if they are not in order (order may be ascending or descending).

Bubble sort algorithm starts by comparing the first two elements of an array and swapping if necessary, i.e., if you want to sort the elements of array in ascending order and if the first element is greater than second then, the elements are swapped but, if the first element is smaller than second, elements are not swapped. Then, again second and third elements are compared and swapped if it is necessary and this process continues until last and second last element are compared and swapped. This completes the first iteration of bubble sort.

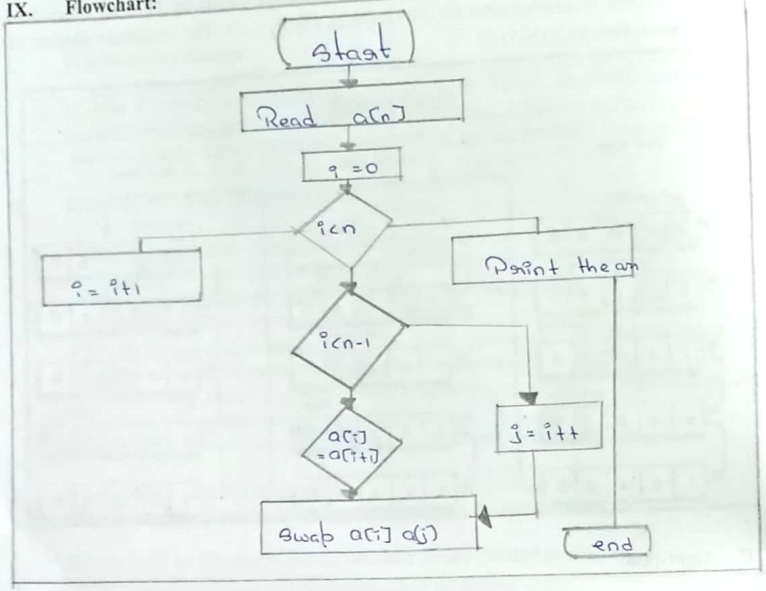
In general it may be required to go through maximum passes as (N-1) for every pass the number of comparisons are (N-1), (N-2), (N-3), ..., 1. The maximum number of comparisons are N(N-1) / 2.



VIII. Algorithm:

1. start
2. Declare element array.
3. if a[i] > a[i+1]
   temp = A[i]
   A[i] = temp
4. Repeat step (3) till n-1
5. Display stored list element.
6. Stop.

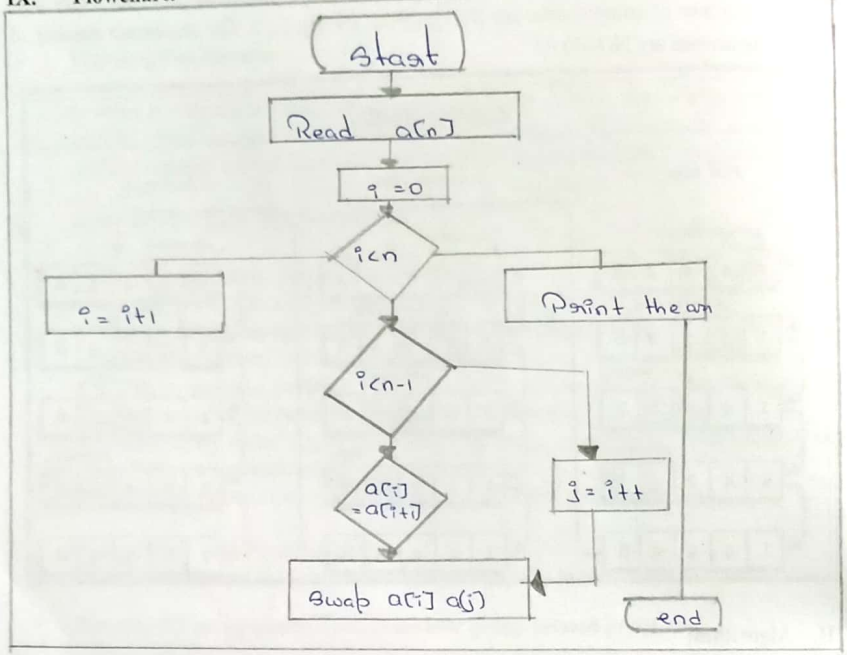
IX. Flowchart:



X. C Program Code:


```
~~main()~~
# include <stdio.h>
# include <conio.h>

void bubble shot (int x[] ,int n)
{
    int i, j, temp;
    for( i=0; i<=n; i++)
    {
        for( j=0; j<=n-1; j++)
        {
            if (x[i] > x[j])
            {
                temp = x[i];
                x[i] = x[i+1];
                x[i+1] = temp;
            }
        }
main()
{
    int i,n,x,~~x~~[n]
clrscr();
        printf (" enter element");
        scanf (" %d", &n);
        printf (" Data \n");
    for( i=0; i<n ; i++)
        scanf ("%d", &x[i]);
    bubble sort (x,n);
    getch ();
}
```

IX. **Flowchart:**



X. **C Program Code:**

XI. **Resources required**

Sr. No.	Name of Resource	Specification	Quantity	Remarks
1	Hardware: Computer System	Computer (i3-i5 preferable), RAM minimum 2 GB and onwards	As per batch size	For all Experiments
2	Operating system	Windows 7 or Later Version/LINUX version 5.0 or Later Version		
3	Software	Turbo C /C++ Version 3.0 or Later Version		

XII. **Precautions**

1. Save the program in specific directory / folder.
2. Follow safety practices.

XIII. **Resources used**

S. No.	Name of Resource	Specification
1	Computer System with broad specifications	Core i3 4GB RAM
2	Software	Turbo C++
3	Any other resource used	Window 7

XIV. **Results (Output of the Program)**

Enter the element the array 30,40,
50,80,10
The Sorted array element are
10,30,40,50,80

XV. **Conclusion(s)**

Medium in bubble sort the idea of the algoritm is to move the higher right and love value element to toward left.

(Space for answers)

1) list does not find list of 5 index.

2)

0	1	2	3	4
-20	50	140	300	1000

3)

0	1	2	3	4
-210	80	120	500	2000

4) Pass-1

1000, -210, 300, 140, 500

-210, 1000, 300, 140 500

-210, 1000, 140, 300, 500

-210, 300, 140, 1000, 500

-210, 300, 140, 500, 1000

-210, 140, 300, 500, 1000

Sorted

XVIII. References / Suggestions for further Reading

1. https://www.w3resource.com/c-programming-exercises/searching-and-sorting/c- search-and-sorting-exercise-3.php (as on 17/01/2018).
2. https://www.sitesbay.com/data-structure/c-bubble-sort (as on 17/01/2018).
3. https://www.programiz.com/dsa/bubble-sort (as on 17/01/2018).
4. https://www.codingbot.net/2013/01/bubble-sort-algorithm-and-c-code.html (as on 17/01/2018).
5. https://www.youtube.com/watch?v=y_Nuui4Qf-k (as on 19/01/2018)

XIX. Assessment Scheme

Performance indicators		Weightage
Process related(10 Marks)		**30%**
1	Debugging ability	20%
2	Follow ethical practices.	10%
Product related (15 Marks)		**70%**
3	Correctness of algorithm	15%
4	Correctness of Program codes	25%
5	Quality of input/output messaging and output formatting	25%
6	Timely Submission of report	5%
7	Answer to sample questions	5%
Total (25 Marks)		**100%**

List of Students /Team Members

1. .....
2. .....
3. .....
4. .....

Marks Obtained			Dated signature of Teacher
Process Related(10)	Product Related(15)	Total(25)	
8	12	20	

Practical No. 5: Program to sort an array using selection sort

I. Practical Significance:

In order to perform some operation on specific data element, the data has to be sorted in data collection, and the system requires following a Sorting method. One of the most commonly used methods is Selection Sort for sorting data from the given list.

II. Relevant Program Outcomes (POs)

- **Basic knowledge:** Apply knowledge of basic mathematics, sciences and basic engineering to solve the computer group related problems.
- **Discipline knowledge:** Apply Information Technology knowledge to solve broad-based Information Technology related problems.
- **Experiments and practice:** Plan to perform experiments and practices to use the results to solve the computer group related problems.
- **Engineering tools:** Apply relevant Computer programming / technologies and tools with an understanding of the limitations.
- **Communication:** Communicate effectively in oral and written form.

III. Competency and Practical skills

This practical expects to develop the following skills in the student.

Develop 'C' programs to solve computer group related problems.

1. Write algorithm and draw flow chart for selection sort.
2. Write / Compile / Debug and execute 'C' program for Selection sort.

IV. Relevant Course Outcome(s)

- Apply different searching and sorting techniques.

V. Practical Outcome (PrOs)

- Implement a 'C' program to sort an array using Selection Sort method.

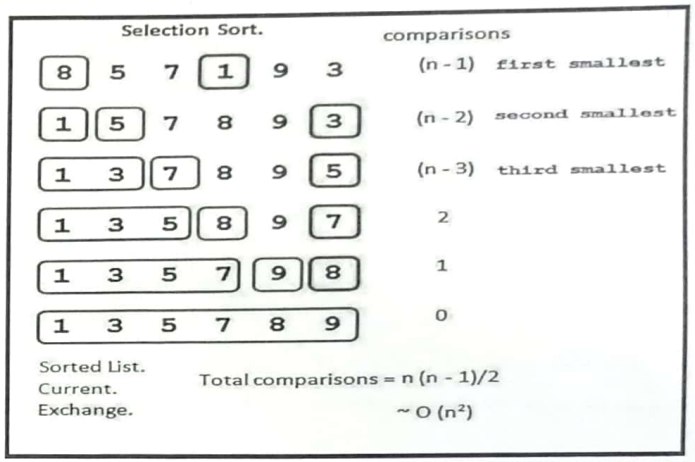
VI. Relevant Affective domain related Outcome(s)

1. Follow safety measures.
2. Follow ethical practices.

VII. Minimum Theoretical Background

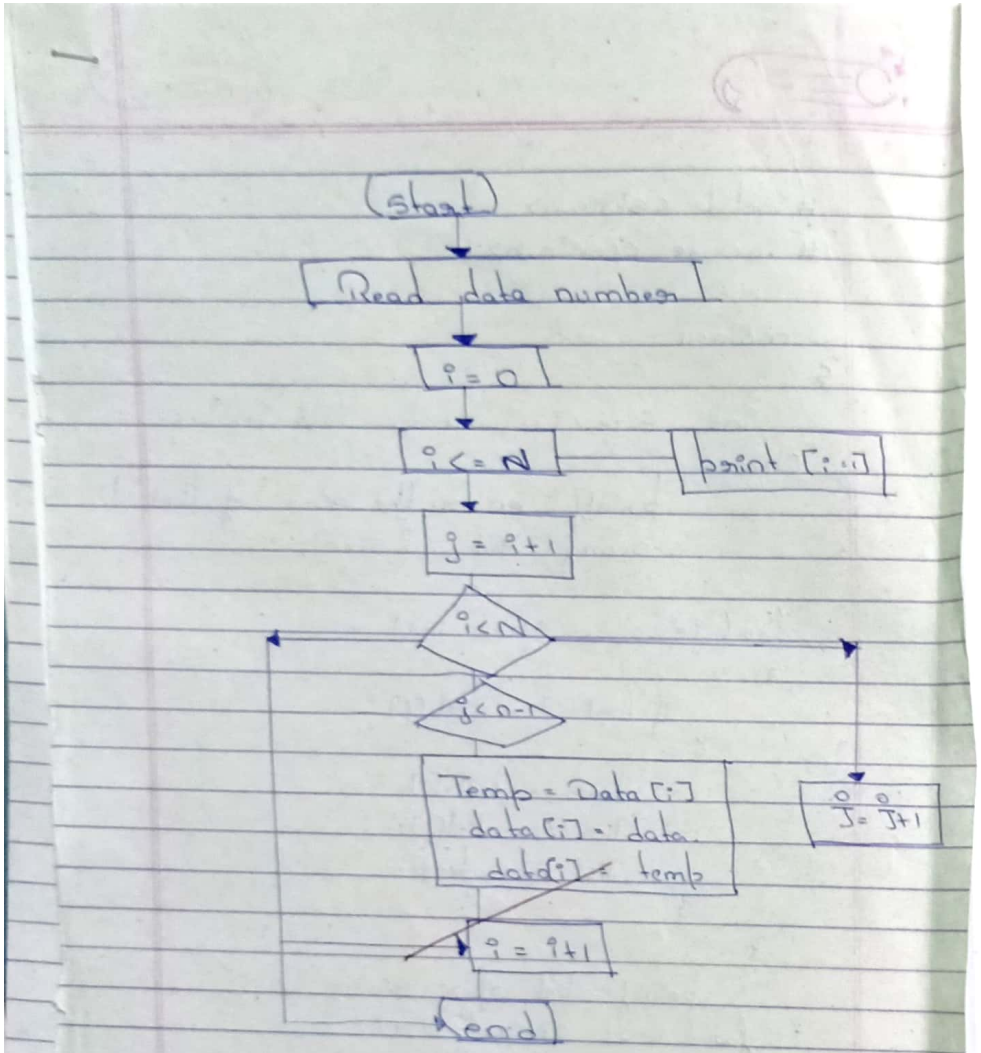
Selection sort is similar to the hand picking where we take the smallest element and put it in the first position and the second smallest at the second position and so on.
We follow the following steps to perform selection sort:

1. Start from the first element in the array and search for the smallest element in the array.
2. Swap it with the value in the first position
3. Repeat the steps above for the remainder of the list (starting at the second position and Advancing each time)



VIII. Algorithm:

1. Start.
2. Repeat step 3 to 5 for i = 1 to n - 1.
3. Min = A[i]
4. A[i] < min
   A[i] = min
   min = temp
   end if n.
5. temp = A[i]
   A[j] = min
   A[n] = temp.
6. End.



XI. Resources required

Sr. No.	Name of Resource	Specification	Quantity	Remarks
1	Hardware: Computer System	Computer (i3-i5 preferable), RAM minimum 2 GB and onwards	As per batch size	For all Experiments
2	Operating system	Windows 7 or Later Version/LINUX version 5.0 or Later Version		
3	Software	Turbo C /C++ Version 3.0 or Later Version		

XII. Precautions

1. Save the program in specific directory / folder.
2. Follow safety practices.

XIII. Resources used

S. No.	Name of Resource	Specification
1	Computer System with broad specifications	Core i3 4GB RAM
2	Software	Turbo C++
3	Any other resource used	Window 7

XIV. Results (Output of the Program)

Enter element to the sorted
enter element 15 enter element 30
enter element 1 enter element 10
in assending order 1,5,10,15,30

XV. Conclusion(s)

Element to the enter element below ~~of~~ of
Sulate multiday to 1,2,4,8

XVI. Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions as to ensure the achievement of identified CO.

Choose the right option from the following:

1. A Selection Sort compares adjacent elements, and swaps them if they are in wrong order.
   a. True b. False c. Depends on Elements d. None of these
   Ans : _____

2. For each i from 1 to n-1, there are _____ exchanges for Selection Sort:
   a. 1 b. n-1 c. n d. None of these
   Ans : _____

3. What is the output of selection sort after the 2nd iteration given the following sequence of numbers: 20 12 10 15 2
   a. 2 15 12 10 20
   b. 2 101 12 15 20
   c. 2 12 10 15 20
   d. None of the above
   Ans : _____

XVII. Exercise

1. Find the number of comparisons required in Selection Sort of the following given list having 5 numbers.

0	1	2	3	4
10	20	30	40	50

2. Sort the given array in ascending order using Selection Sort method and show diagrammatic representation of every iteration of for loop.

0	1	2	3	4
1000	- 20	300	140	50

3. Sort the given array in ascending order using Selection Sort method and show diagrammatic representation of every iteration of while loop.

0	1	2	3	4
500	120	2000	-210	89

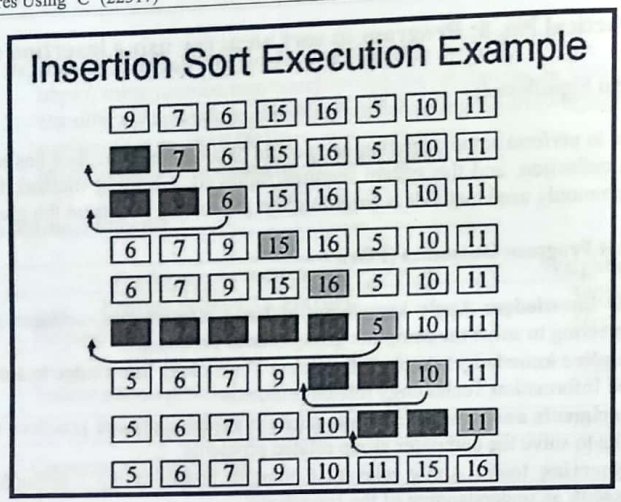
(Space for answers)

2] Pass 1.

500 120 200 -210 89
120 500 200 -210 89
120 200 500 -210 89
120 200 -210 500 89
120 200 -210 89 500

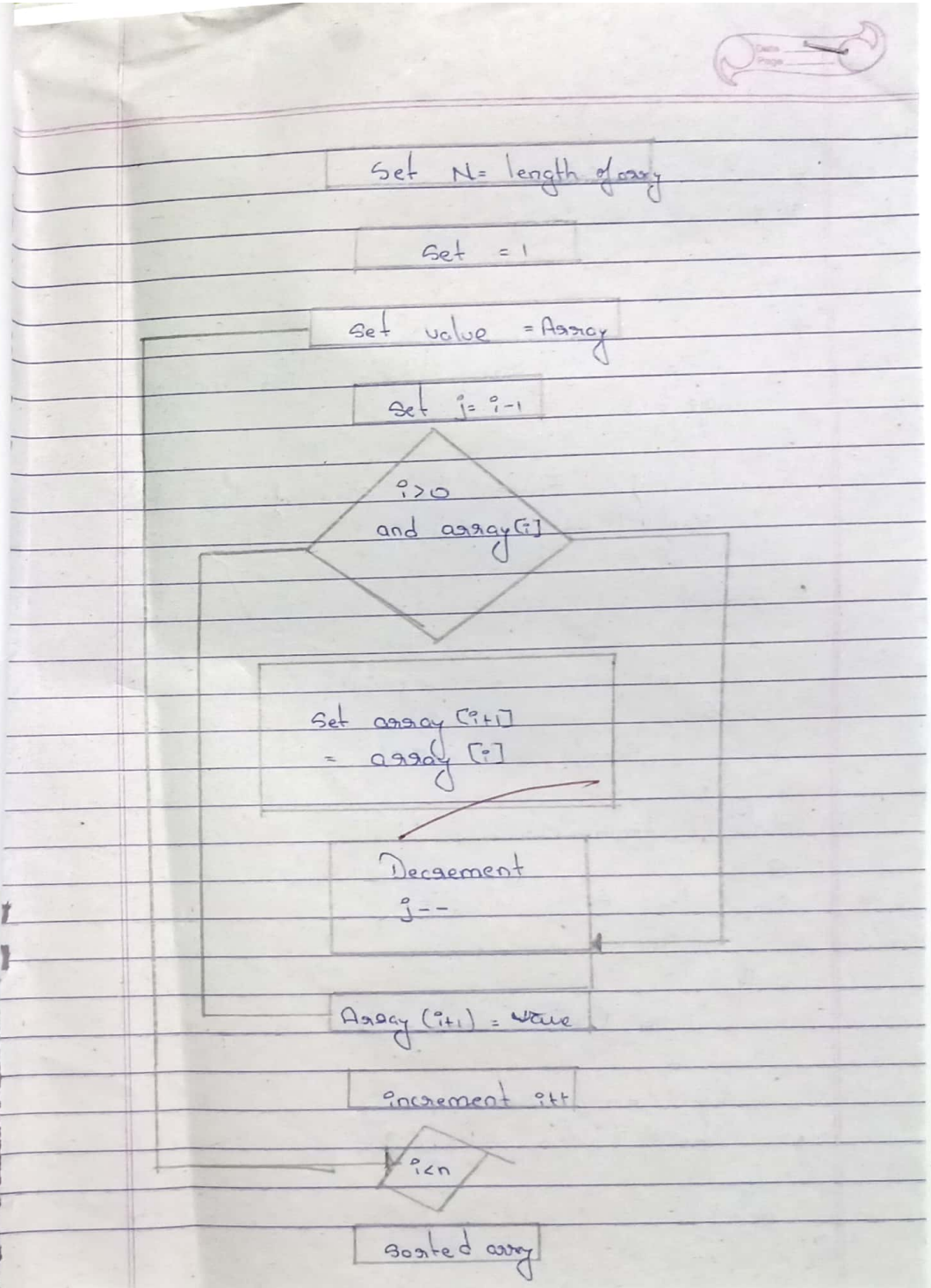
Pass - 2

120 -210 200 89 500
120 -210 89 200 500
-120 89 -120 200 500



VIII. Algorithm:

1. Start
2. Declare N element
3. a[0] Consider a stored file
4. next = a[i]
5. new element = a[next],
6. Move element in array one by position.
7. Insert new element in array out position
8. In formated.
9. Next = next +1.
10. Continue from stop skill next
11. end.



```
# include <stdio.h>
# include <conio.h>

void main()
{
    int i, j, size, temp, list[100];
    clrscr();

    printf(" Enter the size of the list");
    scanf("%d", &size);

    printf("%d", size);

    for(i=0; i<size; i++)
    {
        scanf("%d", a[i]);
    }
while ( temp < list[i] )
{
    list [i+1] = list [i];
    i++;
}
    list [i+1] = temp;

    for( i=0 ; i<size ; i++)
     printf ("%d", list[i]);

    getch();
}
```

XI. Resources required

Sr. No.	Name of Resource	Specification	Quantity	Remarks
1	Hardware: Computer System	Computer (i3-i5 preferable), RAM minimum 2 GB and onwards	As per batch size	For all Experiments
2	Operating system	Windows 7 or Later Version/LINUX version 5.0 or Later Version		
3	Software	Turbo C /C++ Version 3.0 or Later Version		

XII. Precautions

1. Save the program in specific directory / folder.
2. Follow safety practices.

XIII. Resources used

S. No.	Name of Resource	Specification
1	Computer System with broad specifications	Core i3 4GB RAM
2	Software	Turbo C++
3	Any other resource used	Windows 7.

XIV. Results (Output of the Program)

Enter array element = 40, 1, 20, 10, 30, 20, 90

Sorted array = 1, 10, 20, 30, 40, 90

XV. Conclusion(s)

Sorting is done with no errors

XVI. Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

1. Define insertion sort?

Insertion sort is different from other sorting algorithm. Algorithm is sorting from index is compared with element at index is small than element at index a sort during insertion the second element.

2. Differentiate between bubble sort & insertion sort?

Bubble sort	Insertion sort
Bubble sort a technique to arrange by swap	Insertion sorting is a technique for element in a sequence sorting that from by index.
ex- Comparing with index the element a index	The element that index is compared with element at index of first.

Choose the right option from the following:-

3. Insertion sort is 1.in-place 2.out-place 3.stable 4 non-stable
   a.1 and 3 b.1 and 4 c.2 and 3 d.2 and 4
   Ans: 1 and 3

4. Sorting of playing cards is an example of
   a. Bubble sort b. Insertion sort c. Selection sort d. Quick sort
   Ans: Insertion sort

5. Total number of comparisons for insertion sort is
   a.n(n-1)/2
   b. n(n+1)/2
   c.n
   d.n*n
   Ans:_____

XVII. Exercise

1. Find the number of comparisons required in Insertion Sort of the following given list having 5 numbers.

0	1	2	3	4
10	20	30	40	50

2. Sort the given array in ascending order using Insertion Sort method and show diagrammatic representation of every iteration of for loop.

0	1	2	3	4
1000	- 20	300	140	50

3. What is the output of insertion sort after the 2nd iteration given the following sequence of numbers: 7 3 5 1 9 8 4 6

(Space for answers)

1) Ans

Pass 1:

1000 , -20 , 300 , 140 , 50.

-20 , 1000 , 300 , 140 , 50

-20 , 300 , 1000 , 140 , 50

-20 , 300 , 140 , 50 , 1000

-20 , 140 , 50 , 300 , 1000

-20 , 50 , 140 , 300 , 1000

Sorted.

3) 7,3,5,1,9,8,4,6 → 1,3,5,7,8,9,4,6

3,7,5,1,9,8,4,6 → 1,3,4,5,7,8,9,6

3,5,7,1,9,8,4,6 → 1,3,4,5,6,7,8,9

3,5,1,7,9,8,4,6

XVIII. References / Suggestions for further Reading

1. https://www.youtube.com/watch?v=OxN2Jqb8S9s (as on 19/01/2018)
2. https://stackoverflow.com (as on 18/01/2018)
3. https://www.sitesbay.com/data-structure/c-insertion-sort (as on 18/01/2018)
4. https://www.geeksforgeeks.org/insertion-sort/

XIX. Assessment Scheme

	Performance indicators	Weightage
	Process related(10 Marks)	**30%**
1	Debugging ability	20%
2	Follow ethical practices.	10%
	Product related (15 Marks)	**70%**
3	Correctness of algorithm	15%
4	Correctness of Program codes	25%
5	Quality of input/output messaging and output formatting	25%
6	Timely Submission of report	5%
7	Answer to sample questions	5%
	Total (25 Marks)	**100%**

List of Students /Team Members

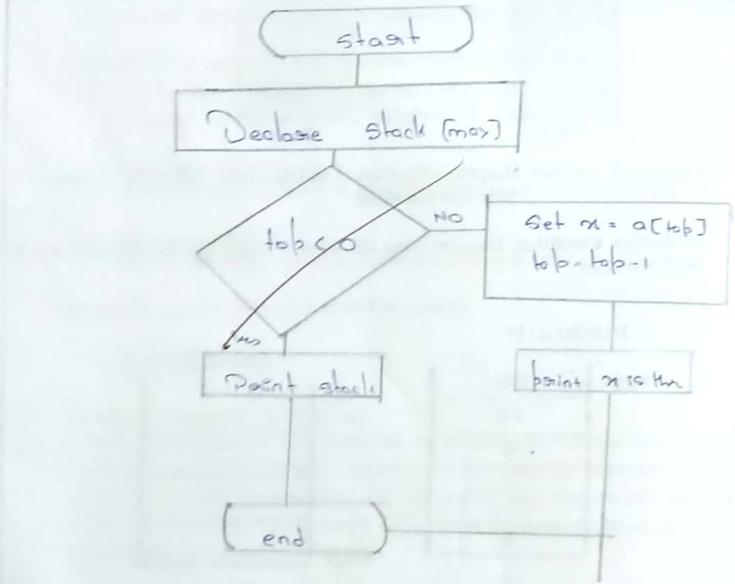
1. .....
2. .....
3. .....
4. .....

Marks Obtained			Dated signature of Teacher
Process Related(10)	Product Related(15)	Total(25)	
8	12	20	5/Feb

VIII. Algorithm

1. Initialize the stack to be empty.
2. Determine whether stack is empty or not
   if(s -> top) if return 1 stack is empty
   else return 0 then condition is false.
3. Determine if stack is full or not if top=max-1
4. Print stack.
5. End.

IX. Flowchart

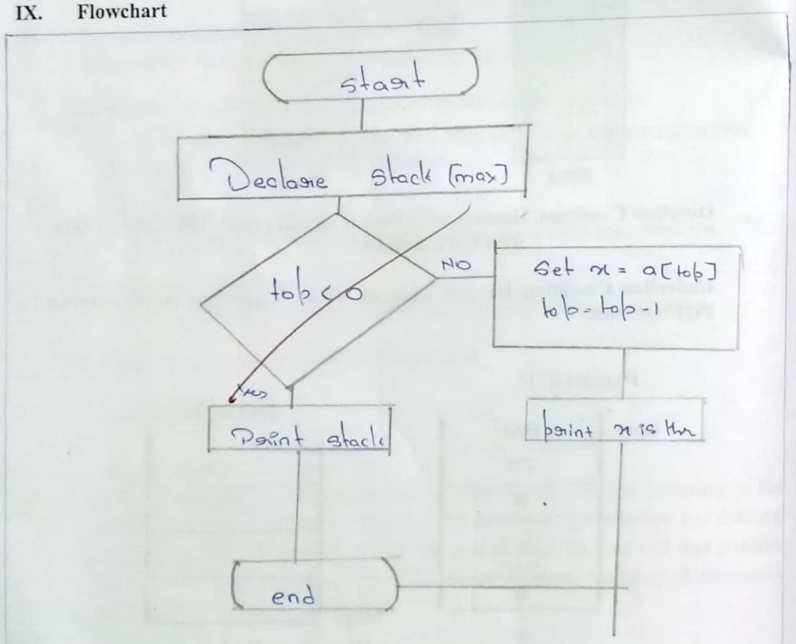



```
switch (ch)
{ case 1: push();
  break;
  case 2: pop();
  break;
  case 3: display();
  break;
  case 4: exit(0);
  break;
  default:
      printf("Wrong Choice");
}
}
void push()
{
  if(top == max-1)
      printf("Stack is full");
  else{
      for(i=0; i<=index; i++)
      {
          a[i+1] = a[i]
      }
```

VIII. Algorithm

1. Initialize the stack to be empty.
2. Determine whether stack is empty or not
   if(s -> top) if return 1 stack is empty
   else return 0 then Condition is false.
3. Determine if stack is full or not if top=no-1
4. Print stack.
5. End.

IX. Flowchart



X. C Program Code

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <process.h>
#define Max 5.
int top = -1  stack [max];

void push();
void pop();
void display();
void main()
{
    int ch;
    Clrscr();
while(1)
{   printf(" stack Menu");
printf(" 1. Push \n 2. POP , 3. Display 4. exit");
scanf(" %d" ,& ch);
```

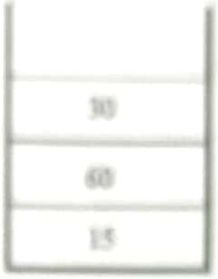
XI. Resources required

Sr. No.	Name of Resource	Specification	Quantity	Remarks
1	Hardware: Computer System	Computer (i3-i5 preferable), RAM minimum 2 GB and onwards	As per batch size	For all Experiments
2	Operating system	Windows 7 or Later Version/LINUX version 5.0 or Later Version		
3	Software	Turbo C /C++ Version 3.0, or later , gcc compiler		

```
    a[index] = element;
}

void pop()
{
    if (top == -1)
        printf("stack is empty");
    else {
        top = top-1;
    }
}

void display()
{
    for (i=0; i<= size; i++)
    {
        printf("%d", a[i]);
    }
}
```

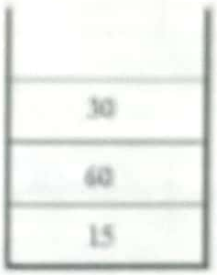
2) Draw stack after PUSH operation using given stack

PUSH(45)



3) Write a TOP value after PUSH operation.

PUSH(45)



XVII. Exercise

(**Note:** Programming exercise use blank pages provided or attach more pages if needed.)

1) Perform following operation on stack of size 5.
   PUSH(30)
   PUSH(12)
   PUSH(10)
   POP()
   PUSH(45)
   POP(70)

2) Implement a "C" program to stack size-8 for push(10), push(20), pop, push(10), push(20), pop, pop, pop, push(20), pop and draw final output.

3) Perform following operation on stack of size 5.
   PUSH(100)
   PUSH(120)
   PUSH(50)
   POP()
   POP()
   POP(70)

(Space for answers)

1) Ans.

Push	Pop	Push
10	12	45
20	30	12
30		30

Push	Pop
70	45
45	12
12	30
30	

2) Ans)

Push	Pop	Push
20	10	20
10		10
		10

Pop	Push	Pop
stack is empty	20	stack is empty

3) Ans)

Push	Pop	Push	Pop
50			
120	100	70	100
100		100	

XVIII. References / Suggestions for further Reading

1. www.geeksforgeeks.org/implement-two-stacks-in-an-array (as on 18/01/2018)
2. www.programmingunit.com/2013/01/14/stack-using-array-c-program (as on 18/01/2018)

XIX. Assessment Scheme

	Performance indicators	Weightage
	Process related(10 Marks)	**30%**
1	Debugging ability	20%
2	Follow ethical practices	10%
	Product related (15 Marks)	**70%**
3	Correctness of algorithm	15%
4	Correctness of Flow chart	15%
5	Correctness of Program codes	20%
6	Quality of input/output messaging and output formatting	5%
7	Timely Submission of report	5%
8	Answer to sample questions	10%
	Total (25 Marks)	**100%**

List of Students /Team Members

1. .....
2. .....
3. .....
4. .....

Marks Obtained			Dated signature of Teacher
Process Related(10)	Product Related(15)	Total(25)	
8	13	21	Chirag

```
#include <stdio.h>
#include <conio.h>
#define size 10.

void InQueue();
void OutQueue();
void display();
int queue[size], front = -1, rear = -1;

void main()
{
    int value, choice;
  Clrscr();
  while (1) {

    printf("\n **Menu **");
    printf(" 1. Insertion ,2. Deletion 3. Display
            4. exit");
    printf(" enter your choice");
    scanf("%d", &choice);

  Switch (choice)
  {
      Case1 :  enque();
        break;

      Case 2 :  dequeue();
        break;
```

```
Case 3: display();
        break;

Case 4: exit(0);

default :
        printf(" enter Correct choice");
}
}
}

void queue
{   if( rear == size-1)
    printf("\n Queue is full");
else{
        if( front == -1)
        { front = 0;
          rear++;
}       queue[rear] = value;
void deque.
  printf(" Queue is empty

else{ printf("\n deleted %d", queue(front));
      front++;
      front = rear-1;
}
```

XII. Precautions

1. Programs should be save in desired location.

XIII. Resources used

S. No.	Name of Resource	Specification
1	Computer System with broad specifications	Ram 4GB Core i3
2	Software	Turbo C++
3	Any other resource used	

XIV. Result (Output of the Program)

.....

.....

.....

.....

XV. Conclusion(s)

.....

.....

.....

.....

XVI. Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

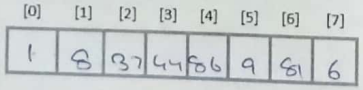
(Note: Programming exercise use blank pages provided or attach more pages if needed.)

1) Write a front and rear values of linear queue.

Enqueue Dequeue

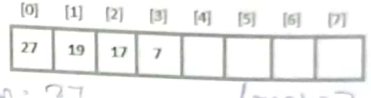
Ans:- 5 rear 7 front.

2) Sketch the front and rear in empty queue.



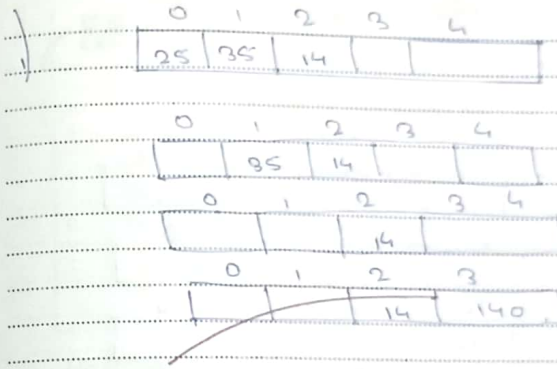
Ans:- Rear = 1 front = 6.

3) Sketch the front and rear in this queue.



Ans:- Rear = 27 front = 7.

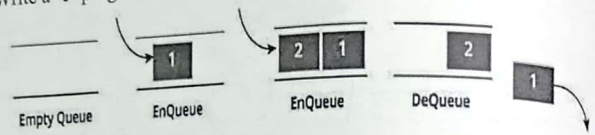
(Space for Answer)



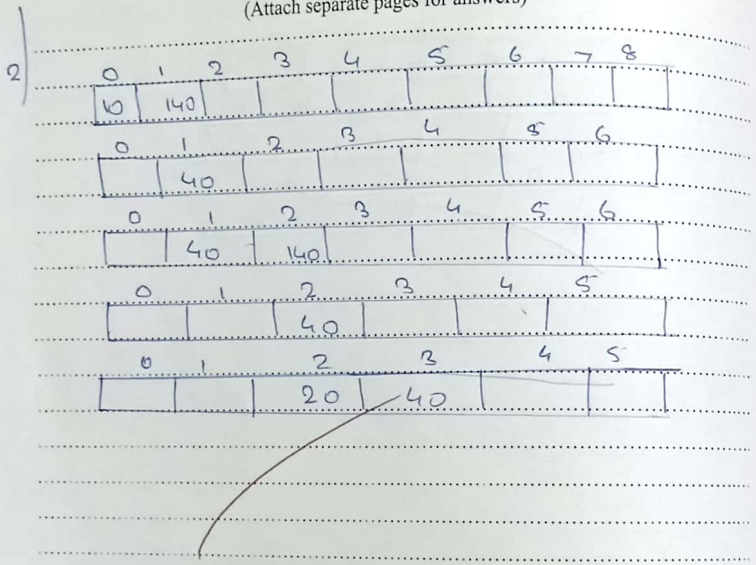
XVII. Exercise

(Note: Programming exercise use blank pages provided or attach more pages if needed.)

1) Implement a C program to Linear queue in array size 5 with this operations INSERT(25), INSERT(55), INSERT(14), DELETE, DELETE, INSERT(40).
2) Implement a C program to Linear queue in array size 10 with this operations INSERT(10), INSERT(40), DELETE, INSERT(40), INSERT(20), DELETE, INSERT(40).
3) Write a 'c' program for linear queue using array from given data.



(Attach separate pages for answers)



XVIII. References / Suggestions for further Reading

1 www.uobabylon.edu.iq/eprints/publication_5_3108_1456.pdf(as on 18/01/2018)
2 www.w3schools.in/data-structures-tutorial/queue/ (as on 18/01/2018)

XIX. Assessment Scheme

	Performance indicators	Weightage
	Process related(10 Marks)	**30%**
1	Debugging ability	20%
2	Follow ethical practices.	10%
	Product related (15 Marks)	**70%**
3	Correctness of algorithm	15%
4	Correctness of Program codes	25%
5	Quality of input/output messaging and output formatting	20%
6	Timely Submission of report	5%
7	Answer to sample questions	5%
	Total (25 Marks)	**100%**

List of Students /Team Members

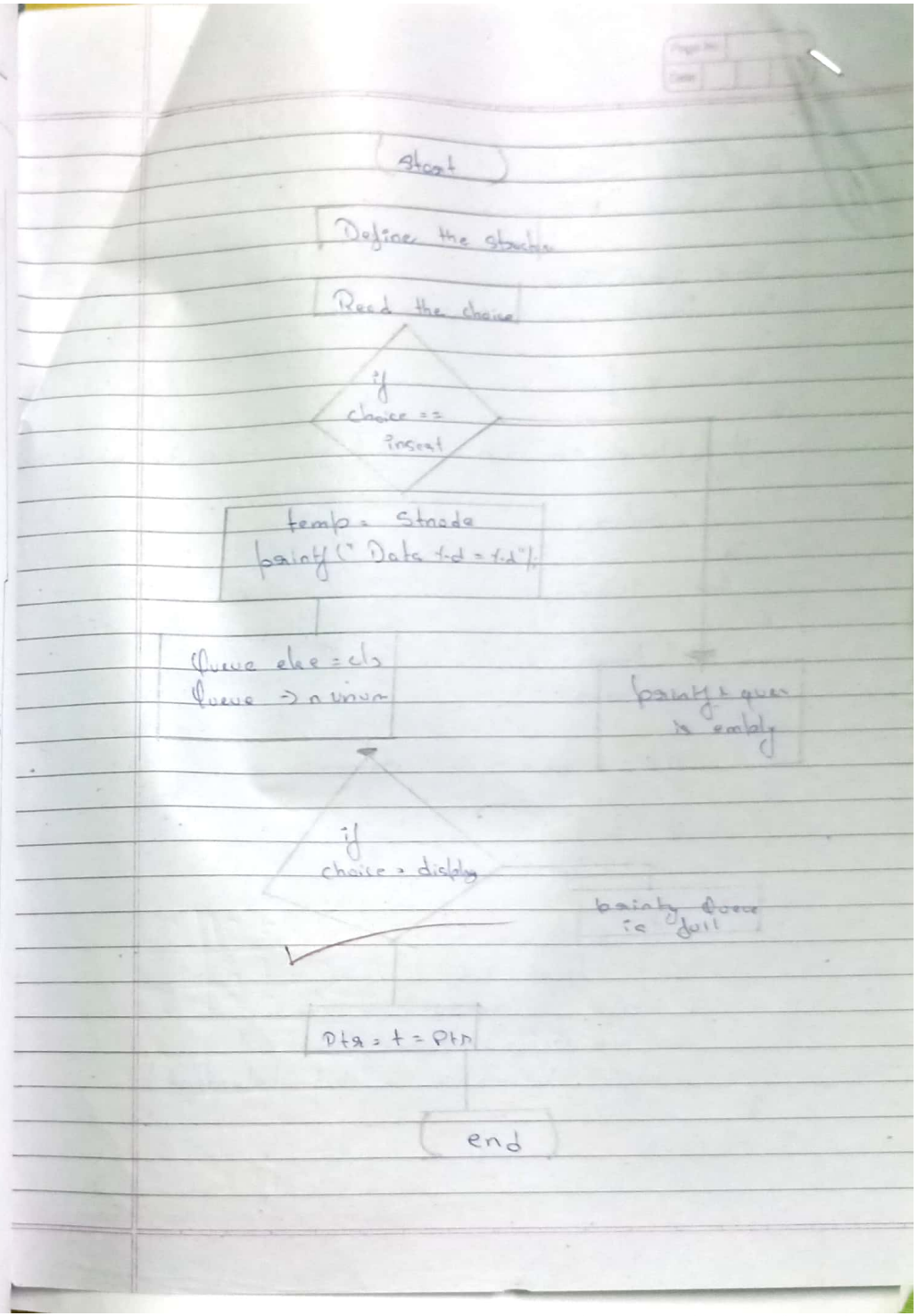
1. .....
2. .....
3. .....
4. .....

Marks Obtained			Dated signature of Teacher
Process Related(10)	Product Related(15)	Total(25)	
8	13	21	

VIII. Algorithm

* Steps :-

1. if (front == (Rear + 1) % max then.
2. printf(" Queue is overflow as exit");
3. else take the value.
4. if (front = rear = 0)
   rear = (rear + 1) % max.
5. Assign value.
   Queue [rear] = value.
6. end if
7. exit.
8. if (front => 0)
9. Printf(" Queue is overflow of exit");
10. else
    element = Queue [front].
11. if (front == -1)
    rear = -1
12. else.
    front = front + % max.
13. end.
14. exit.



```
#include <stdio.h>
#include <conio.h>

int item[size];
int front = -1, rear = -1

int isfull()
{
    if ((front == rear+1) || (front == 0 && rear == size-1))
    {
        return 1;
    }
    int isempty()
{
    if (front == -1)
        return 1;
}
    void enqueue (int element)
{
    if (isfull())
    printf("\n Queue is full \n");
else {
        if (front == -1)
            front = 0;
        rear = (rear+1) % size;
        item[rear] = element;
    printf("\n location = %d, element);
    }
}
```

XI. Resources required

Sr. No.	Name of Resource	Specification	Quantity	Remarks
1	Hardware: Computer System	Computer (i3-i5 preferable), RAM minimum 2 GB and onwards	As per batch size	For all Experiments
2	Operating system	Windows 7 or later Version/LINUX version 5.0 or later Version		
3	Software	Turbo C /C++ Version 3.0, or later , gcc compiler		

XII. Precautions

1. Save program in dedicated folder.

XIII. Resources used

S. No.	Name of Resource	Specification
1	Computer System with broad specifications	Core i3 4 GB RAM.
2	Software	Turbo C++
3	Any other resource used	Window 7.

XIV. Result (Output of the Program)

Circular Queue is a linear data structure in which the operation are program become possible and last

XV. Conclusion(s)

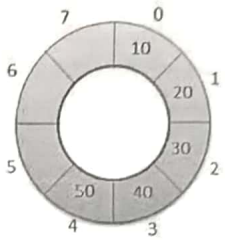
A Circular Queue can be clib be as fellow in linear data output in which the operation

XVI. Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

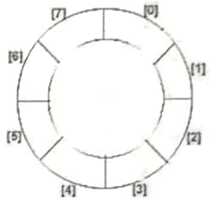
(Note: Programming exercise use blank pages provided or attach more pages if needed.)

1) Write a front and rear values of circular queue.



Ans: Front = 10
Rear = 50

2) Show the front and rear in empty circular queue.

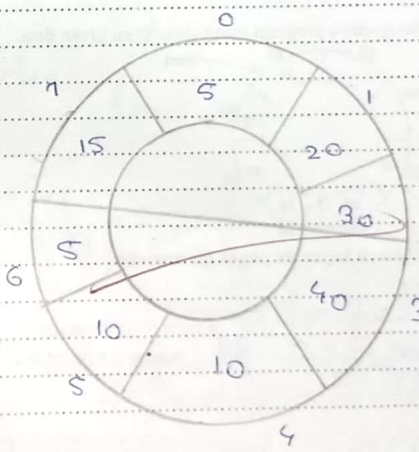


Ans: Front = -1
Rear = -1.

3) Show the circular queue using given values of array (size is 5).
15, 20, 30,10,5

Ans:-

(Space for Answer)

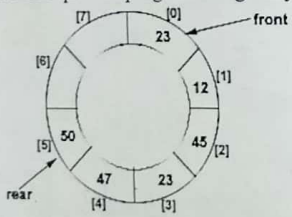


XVII. Exercise

Attempt Q1. and teacher shall allot Q. 2/Q.3 from the following:

(Note: Programming exercise use blank pages provided or attach more pages if needed.)

1) Implement a C program to circular queue in array size 5 with this operations INSERT(25), INSERT(55), INSERT(14), DELETE, DELETE, INSERT(40).
2) Implement a C program to circular queue in array size 10 with this operations INSERT(10), INSERT(40), DELETE, INSERT(40), INSERT(20), DELETE, INSERT(40).
3) Write a circular queue c program using array from given data.



(Attach separate pages for answers)

```
void insert (int x) {
    if ((front == 0 && rear == MAX-1) || rear == front-1)
        printf("\n Queue is Overflow");
```



```
    else {
        if (rear == -1)
            rear = front = 0;
        else
            rear ++;
        a[rear] = x;
        printf("\n %d is inserted", x);
    }
}
void delete () {
    int delItem;
    if (front == -1)
        printf("\n Queue is underflow");
    else {
        delItem = a[front];
        printf("\n %d is deleted from Queue",
        delItem);
        if (front == MAX-1)
            front = 0;
        else
            front ++;
    }
}
void display () {
    int i;
    if (front <= rear) {
        for (i=front; i<=rear; i++)
            printf("%d", a[i]);
    }
}
```

VII. Minimum Theoretical Background

Linked List: Linked List data structure consists of collection of nodes in a sequence which is divided in two parts. Each node consists of its own data and the address of the next node and forms a chain. Linked Lists are used to create stacks, queues,trees and graphs.

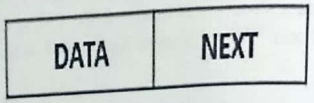


Fig1. Node

Data holds the data variable while Next holds the address to the next Node in the list.

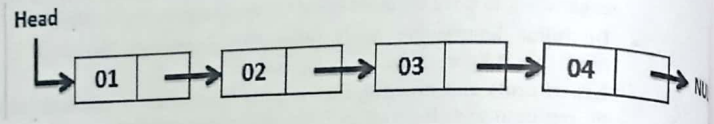


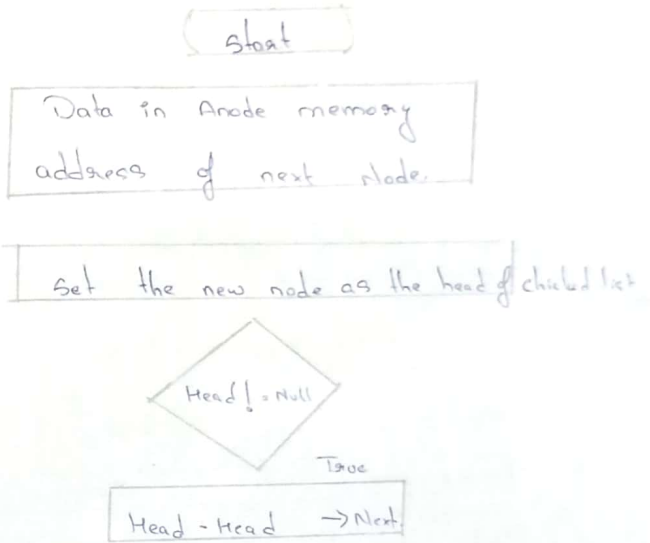
Fig 2. Linked List

Head is a pointer variable of type struct Node which acts as the Head(starting of a node) to the list. Initially we set 'Head' as NULL which means list is empty. Basically Single Linked Lists are uni-directional as they can only point to the next Node in the list but not to the previous. The operations we can perform on singly linked lists are insertion, deletion and traversal.

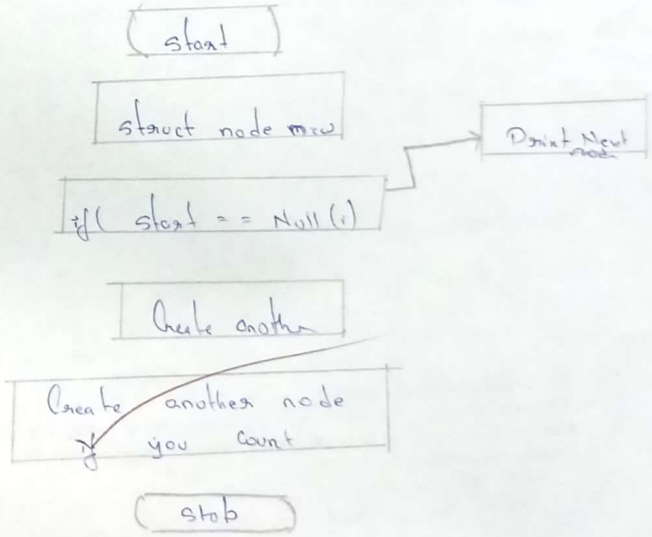
VIII. Algorithm (Attach separate paper if required)

1. **Insert a node**
   - i. At Beginning of Linked List
   - ii. At the end of Linked List
   - iii. At a given position in Linked List
2. **Delete a node**
   - i. At Beginning of Linked List
   - ii. At the end of Linked List
   - iii. At a given position in Linked List
3. **Traversing Linked List**
4. **Searching data from Linked List**

IX. Flow Chart (Attach separate paper if required)



X. 'C' Program Code(Attach separate paper if required



XI. Resources required

Sr. No.	Name of Resource	Specification	Quantity	Remarks
1	Hardware: Computer System	Computer (i3-i5 preferable), RAM minimum 2 GB and onwards	As per batch size	For all Experiments
2	Operating system	Windows 7 or Later Version/LINUX version 5.0 or Later Version		
3	Software	Turbo C /C++ Version 3.0 or later		

XII. Precautions

1. Be careful while giving input for dynamic memory allocation.
2. Follow pointer's fundamentals.

XIII. Resources used

S. No.	Name of Resource	Specification
1	Computer System with broad specifications	
2	Software	
3	Any other resource used	

XIV. Result

.....
.....
.....
.....

XV. Conclusion(s)

.....
.....
.....
.....

XVI. Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

Choose right option for following:

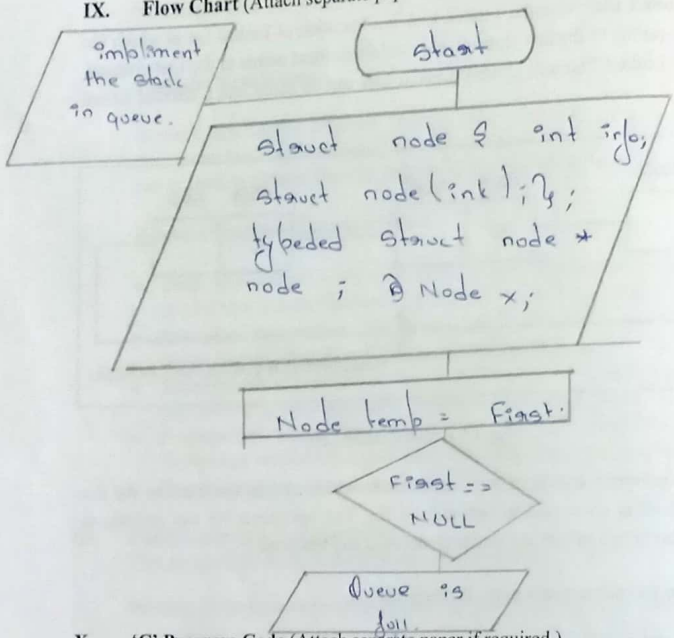
1. Single linked list uses _____ no. of pointers

   a. Zero **b.** one **c.** Two **d.** Three

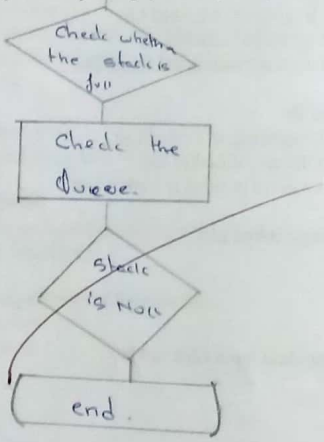

```
# include <stdio.h>
# include <conio.h>
  struct node
{
    int num;
// the node
  struct node * long nextptr;  // Adress
} * stnode;

void Create Node list (int n). // function
Create the list
void display list();
display the list
    (int main()
{
   int n;
   printf (" linked list : to Create and display
   Single linked list : \n");

   printf (" Input the number of nodes:");
   scanf ("%d", &n);
   Create Node list (n);
   printf ("\n Data entered in list : \n");
   display list ();
   return 0;
}
   void Create Node list (int n);
}
```

```
struct node * Fn node ** temp;
int num, i;
Snode = (struct node*) malloc (size of
                (struct node));
if (snode == Null) // check whether the
fnnode is Null and if so no memory
allocation.
{
   printf (" Memory Can not be Allocated");
}

else{
   printf (" Put data for node %:");
   scanf ("%d", &no);
   scanf ("%d", &num);

   st node -> Num = num;
   snode -> Next ptr = Null;
   address field to Null
   temp = Snode;
{
   fn node = (struct node *) malloc
   (size of (Structnode));

   if (FN Node == Null),
{
   printf (" Memory Can, not be allocated");
   break;
}
{ printf (" input data for node %d", i);
   scanf ("%d", &num);
   fn-no -> Num = num;
   field of fnnode with num
fn node -> Nestptr = Null;

field of fnNode with Null.
```

IX. Flow Chart (Attach separate paper if required)



X. 'C' Program Code (Attach separate paper if required.)



XI. Resources required

Sr. No.	Name of Resource	Specification	Quantity	Remarks
1	Hardware: Computer System	Computer (i3-i5 preferable), RAM minimum 2 GB and onwards	As per batch size	For all Experiments
2	Operating system	Windows 7 or Later Version/LINUX version 5.0 or Later Version		
3	Software	Turbo C /C++ Version 3.0 or later		

XII. Precautions

1. Be careful while giving input for dynamic memory allocation.
2. Keep track of next pointer to observe Circular singly linked list.

XIII. Resources used

S. No.	Name of Resource	Specification
1	Computer System with broad specifications	Core i3 4GB RAM
2	Software	Turbo C++
3	Any other resource used	

XIV. Result (Output of the Program)

.....

.....

.....

.....

XV. Conclusion(s)

.....

.....

.....

.....

XVI. Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

Choose the right option from following:

1. Circular Single linked list uses _____ no. of pointers

   a. Zero **b.** one **c.** Two **d.** Three

2. No. of pointers to be manipulated in a Circular linked list to insert an item in the middle _ _
   a. Two **b.** Three **c.** One **d.** Zero
3. Linked lists are not suitable for data structures of which one of the following problem?
   a. insertion sort **b.** Binary search **c.** radix sort
   d. polynomial manipulation problem
4. The last node of Circular linked list field(s) having
   a. data **b.** pointer **c.** pointer to next node **d.** pointer to first node

XVII. Exercise

1. Give the benefit of Circular Single Linked List.
2. Represent Linked List as Circular Queue.

(Space for answers)

1) Ans:)
A Circular linked has no being and his end.

-> We can converse a circular list in any direction either forward and or backward.

-> At some when have to go to the first node from the last node.


```
# include <stdio.h>
# include <conio.h>

struct node
{
    int num;
    struct node *nextptr;
} *stnode;

void CList Creation (int n);
void display (list ();
int main()
{
    int n;
    stnode = Null;

printf(" Circular linked list");
Created and display a Circular linked
list :\n");

printf("
                                \n");
printf(" input the number of nodes:");
scanf ("%d", &n);
CList Cration (n);
display list ();
return 0;
}
void ClList Crection (int n).
{
```

```
int i, num;
struct node * preptr * newnode;
if ( n >=1),

{  stnode = (struct node*) malloc (size of
            (struct node));

printf (" Input data for node 1:");
scanf (" %d", &num);

stnode -> num = num;
stnode => nextptr = NULL;
preptr = stnode.

for(i=2; ic=n; i++)
{
   newnode = (struct node*) malloc (size of
    (struct node));
   printf (" %d
   preptr = newnode ; Previous node is
   advanced.
}

Preptr -> Nextptr = stnode // last node is
linking with first node.
}
}

void display (list c)
{
```

```
struct node * temp;
int n=1;
if ( stnode == NULL)
{
   printf (" No Data found in the list yet");
}

else
{
   temp = stnode;
   printf (" \n\n Data entered in the list are:");

   {
      printf (" Data %d \n", n, temp -> num);

   tmp = temp -> nextptr;
   n++;
}
```

Practical No. 12: Perform traversing on binary search tree

I. Practical Significance

To perform operations in some application which works on hierarchical data a Tree data structure is most known Data structure. Tree is used to represent data in hierarchical manner. A binary tree can be used to represent ordered array and a linked list.

II. Relevant Program Outcomes (POs)

- **Basic knowledge:** Apply knowledge of basic mathematics, sciences and basic engineering to solve the broad-based Computer engineering problem.
- **Discipline knowledge:** Apply Information Technology knowledge to solve broad-based Information Technology related problems.
- **Experiments and practice:** Plan to perform experiments, practices and to use the results to solve Information Technology related problems.
- **Engineering tools:** Apply appropriate Computer Engineering / Information Technology related techniques/ tools with an understanding of the limitations.
- **Communication:** Communicate effectively in oral and written form.

III. Competency and Practical skills

This practical is expect to develop the following skills in you

Develop 'C' programs to solve broad-based computer group related problems.

1. Write algorithm and draw Flow Chart for Traversing Tree.
2. Write a C program for Traversing Tree
3. Compile/Debug/Save the C program.

IV. Relevant Course Outcome(s)

- Implement program to create and traverse tree to solve problems.

V. Practical Outcome

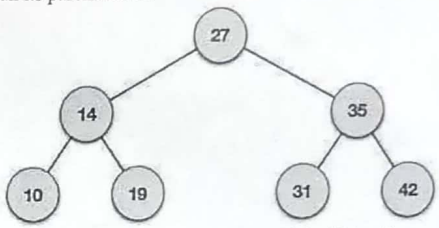
- Write C program to implement BST(Binary Search Tree) and traverse the tree (Inorder, Preorder, Postorder)

VI. Relevant Affective domain related Outcome(s)

1. Follow safety measures
2. Follow ethical practices.

VII. Minimum Theoretical Background

Binary Search Tree (BST) : A binary search tree is a tree where each node has a left and right child. Either child, or both children, may be missing. A node's left child must have a value less than its parent's value and the node's right child must have a value greater than its parent value.



Traversing : Unlike linear data structures (Array, Linked List, Queues, Stacks, etc) which have only one logical way to traverse them, trees can be traversed in different ways. Following are the generally used ways for traversing trees.

Tree Traversals:

(a) Inorder
(b) Preorder
(c) Postorder

VIII. Algorithm (Attach separate paper if required)

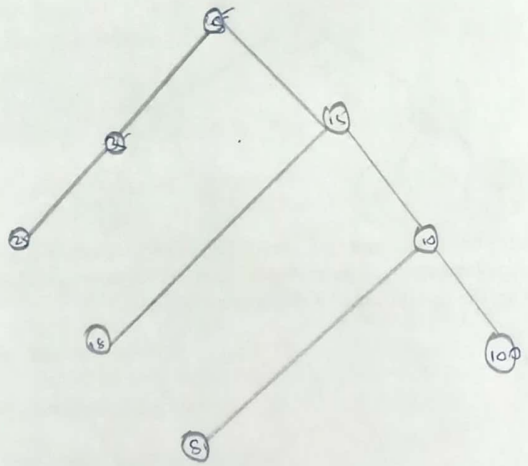
1. Create Binary Search Tree

Step 1: Set NEW = Create a new node.
Step 2: Set NEW = INFO = ITEM.
Step 3: Set NEW = LEFT = NULL AND
Set NEW = RIGHT = NULL.
Step 4: IF (ROOT = NULL) then
Set ROOT = NEW
Else If (ITEM < PTR -> INTRO)
then Set PTR -> LEFT = NEW

Step 5: Repeat steps 1 to 4 while ch = N OR ch = n.

2. Traversal

i. Inorder
ii. Preorder
iii. Postorder

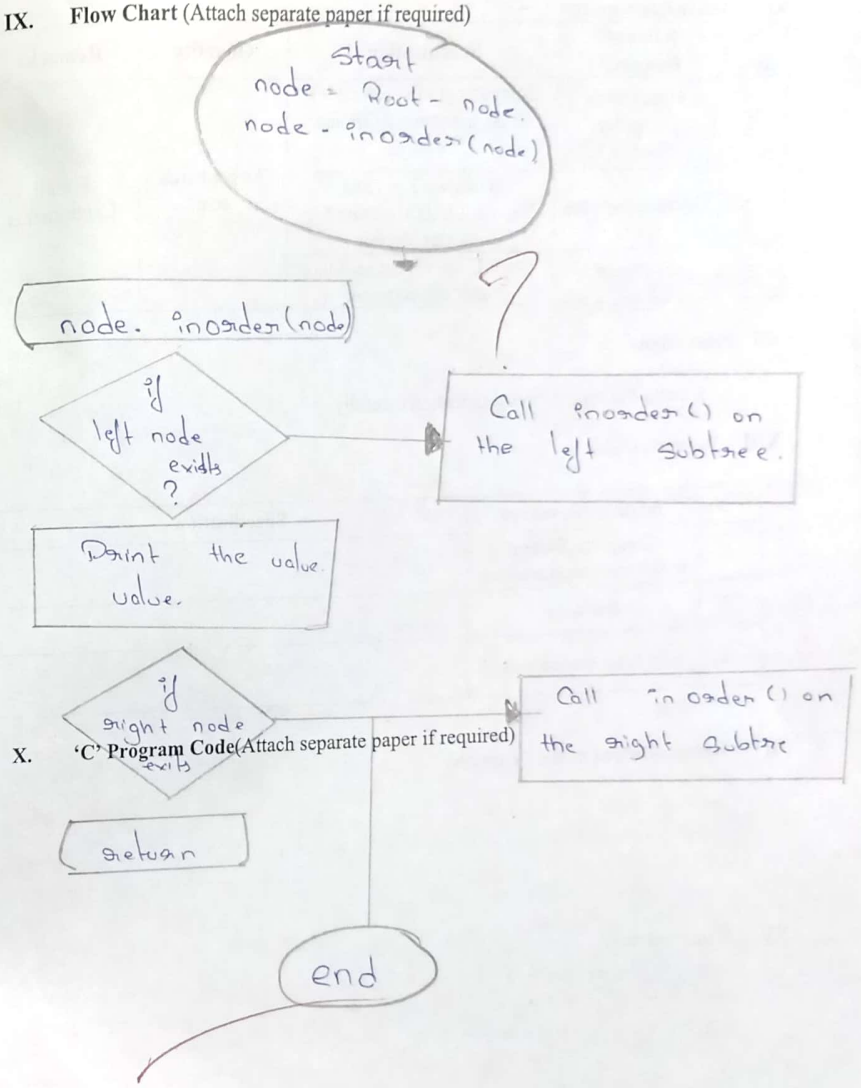


In Order - 25, 35, 45, 18, 15, 31, 20, 40

Preorder - 45, 35, 25, 15, 18, 20, 31, 40.

Post order - 25, 18, 35, 31, 40, 20, 15, 45.

IX. Flow Chart (Attach separate paper if required)



X. 'C' Program Code (Attach separate paper if required)

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

struct node{
 int key;
  struct node *(left) * right;
};
  struct node * newnode (int item)
{
  struct node * temp.
= (struct node *) malloc (size of (struct
                          node));
 temp -> key = item;
 temp -> left = temp -> Right = Null;

Return temp;
}

 void inorder (struct node * Root)
{
   if( Root != Null)
  {
    in order (Root -> left);
   printf(" %d" , Root -> key);
   inorder (Root -> Right);
  }
}
```

```
struct node * insert (struct node
  node, int key)
{
    if (node == Null)
  return new node (key);
  if (key < node -> (key);
  node -> left = insert (node -> Right,
  return node;
}

  int main()
{
  /*  50
     /  \
    30    70
   /  \   /  \
  20  40 60  80   */

  struct node * root = Null;
    Root = insert (Root, 50);
    insert (Root, 30);
    insert ( Root, 20);
    insert ( root, 40);
    insert ( root, 70);
    insert ( root, 60);
    insert ( root, 80)
}   return 0;
```