Index

Sl No.	Name of the Experiment	Page No.	Date of Experiment	Date of Submission	Remarks
1	static initialization of array	2			
2	update array	2			
3	variable length input in array	3			
4	Dynamic input array (Student Database)	4			
5	Linear Search	5			
6	Binary Search	6			20/10/22
7	Create First (singly linked list)	7			
8	beginInsert (singly linked list)	8			
9	deleteFirstNode (singly linked list)	9			
10	insertNodeBeforeVal (singly linked list)	10			
11	search value (singly linked list)	11			
12	traverse (singly linked list)	12			
13	Remove duplicate element (Linked list)	13			
14	Find & display middle elem (Linked list)	14			
15	Count & display repeated elem (Array)	15			20/10/22
16	Push & Pop (stack : array)	16			
17	Push & Pop (stack : linked list)	17			
18	Linear Queue (array)	18			
19	Linear Queue (linked list)	20			
20	Circular Queue (array)	22			
21	Bubble Sort	24			
22	Insertion Sort	25			
23	Selection Sort	26			
24	Heap Sort	27			
25	Maximum & minimum in array	29			
26	Concatenate two arrays	30			

Index

Sl No.	Name of the Experiment	Page No.	Date of Experiment	Date of Submission	Remar
27	Delete last node (linked list)	31			
28	Delete particular value (linked list)	32			
29	Tree traversal	33			
30	Insertion & searching in BST	34			

① ⇒ Static Initialization of Array

3 5 7 9 11

② ⇒ Update value at specific position in array

Old array is :

2 4 6 8 10

Enter the index which you want to update: 3

Enter the value you want to set at index 3 : 5

New array is :

2 4 6 5 10

1> Write a program to show the static initialization of array elements.

```
# include <stdio.h>
int main() { printf ("Static Initialization of Array\n");
        int arr[5] = { 3, 5, 7, 9, 11};
        for (int i=0 ; i<5 ; i++) {
                printf ("%d ", arr[i]);
        }
return 0;}
```

2> Write a program to update the value at specific position in array.

```
# include <stdio.h>
int main() { printf (" Update value at specific position in array\n");
        int arr[5] = {2, 4, 6, 8, 10};
        printf (" Old array is : \n");
        for (int i=0 ; i<5 ; i++) {
                printf ("%d. ", arr[i]);
        }
        int index, value;
        printf ("\n Enter the index which you want to update: ");
        scanf ("%d", &index);
        printf ("\n Enter the value you want to set at index %d:", index);
        scanf ("%d" ,&value);
        arr[index] = value;
        printf ("\n New array is : \n");
        for (int j=0 ; j<5 ; j++) {
                printf ("%d ", arr[j]);
        }
return 0;
```

Output :-

③ ⇒ Variable length input of elements

```
Enter length of array : 4

Enter value at 0 index : 9
Enter value at 1 index : 8
Enter value at 2 index : 7
Enter value at 3 index : 6
The array is
9 8 7 6
```


(3) Write a program to show the variable length input of elements in the array.

```
# include <stdio.h>
int main () { printf (" Variable length input of elements \n ");
        int size ;
        printf (" Enter length of array :  ");
        scanf (" %d " , &size );
        int arr [size ] ;
        for (int i = 0 ; i<size ; i++) {
                printf (" \n Enter value at %d index :  ", i );
                scanf (" %d ", & arr [i ]);
            }
        printf ("The array is : \n ");
        for (int j = 0 ; j<size ; j++) {
            . printf (" %d  ", arr [j ] );
            }
        return 0; }
```

Output :-

(4) ⇒ Student database
Enter number of students : 2

Enter details of student 1 :
Enter roll no, marks and attendance of student :
1 92 74

Student 1 details are :
Roll no : 1
Marks : 92
Attendance : 74

Enter details of student 2 :
Enter roll no, marks and attendance of student
3 62 10

Student 2 details are :
Roll no : 3
Marks : 64
Attendance : 10

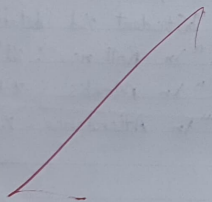
4) Write a program to create a student database dynamic approach of array & display the values according to it.

```
#include <stdio.h>
int main() { printf("Student database \n");
    // int num, roll, marks, attendance;
    int num, arr[3];
    printf("Enter number of students : "); scanf("%d", &num);
    for (int n=0 ; n<num ; n++){
        printf("\nEnter details of student %d : \n", n+1);
    // for(int i=0 ; i<3 ; i++){
        printf("Enter roll no, marks and attendance of student: ");
        scanf("%d %d %d", &arr[0], &arr[1], &arr[2]);

    //displaying
    // for(int i=0 ;
        printf("\nStudent %d details are : ", n+1);
        printf("\n Roll no. : %d", arr[0]);
        printf("\n Marks : %d", arr[1]);
        printf("\n Attendance : %d", arr[2]);

    }

return 0; }
```

Output :-

(5) ⇒ Concept of linear search
Enter size of array : 3
Enter element at index 0 : 13
Enter element at index 1 : 15
Enter element at index 2 : 17
Enter the number to be searched : 15
15 found at index 1

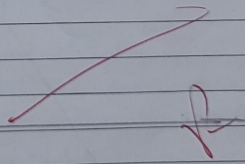
⇒ Concept of linear search
Enter size of array : 3
Enter element at index 0 : 13
Enter element at index 1 : 15
Enter element at index 2 : 17
Enter element to be searched : 89
Element Not Found!



CLASSMATE
Date :
Page : 5

25. Write a program to show the concept of linear search.

```
#include <stdio.h>
int main() { printf("Concept of linear search");
int size, key;
printf("Enter size of array : ");
scanf("%d", &size);
int arr[size];

for(int i=0; i<size; i++){
        printf("Enter element at index %d : ", i);
        scanf("%d, &arr[i]);
    }
    printf("Enter the ~~num~~ value you want to search: ");
    scanf("%d", &key);

for(int i =0; i<size ; i++){
        if(arr[i] != key) { continue; }
        printf("%d found at index %d", key, i);
        return 0;
    }
printf("Element Not Found!");
return -1;
}
```



⑥ ⇒ Concept of binary search
Enter size of array : 3
Enter elements of array :
13
15
17
Enter value to find : 17
17 found at index 2

⇒ Concept of binary search
Enter size of array : 9
Enter elements of array :
2
3
5
7
9
11
13
14
18
Enter value to find : 89
Not found ! 89 is not present

(6) Write a program to show concept of binary search

```
#include <stdio.h>
int main() { printf("Concept of binary search");
    int low, high, mid, n, key;
    printf("Enter ~~number~~ size of array : ");
    scanf("%d", &n);
    int arr[n];

    printf("Enter elements of array : \n");
    for (int i=0 ; i<n ; i++){
            scanf("%d", &arr[i]);
        }

printf("Enter value to find : ");
scanf("%d", &key);

low=0 ; high = n-1 ; mid = (low+high) /2 ;

while (low <= high) {
        if (arr[mid] < key) {
                low = mid + 1 ;
            }
        else if (arr[mid] == key) {
                printf("%d found at index %d \n", key, mid);
        break; }
        else { high = mid -1 ; }
                mid = (low+high) /2 ;
    }
if (low > high) { printf("Not found ! %d is not present ", key);
return 0 ; }
```

Expt. No. 7

Date 14/9/22

Page No. 7

⇒ WAP to create the first node of a linked list.

```
typedef struct SinglyLinkedList {
                    int data;
            struct SinglyLinkedList *next;
    } node;

node* first;

void createFirst (int val) {
        first = (node *) malloc(sizeof(node));
        first -> data = val;
        first -> next = NULL;
    }
```

Date 14/9/22

Expt. No. 8

Page No. 8

⇒ WAP to insert a node at the beginning of a linked list.

```
typedef struct SinglyLinkedList {
            int data;
    struct SinglyLinkedList *next;
} node;

node* first, *temp;

void beginInsert (int val) {
        temp = (node*) malloc (sizeof (node));
        temp → data = val;
        temp → next = first;
        first = temp;
    }
```

Date 14/9/22

Expt. No. 9 Page No. 9

→ WAP to delete the first node of linked list.

```
typedef struct SinglyLinkedList {
                    int data;
        struct SinglyLinkedList *next;
} node;

node* first, *temp;

void deleteFirstNode() {
        temp = first;
        first = first→next;
        temp→next = NULL;
        free(temp);
}
```

Date 14/9/22

Expt. No. 10 Page No. 10

⇒ WAP to add a node before a value in linked list.

```
typedef struct SinglyLinkedList {
                    int data;
        struct SinglyLinkedList *next;
} node;

node* first, *temp, *ptr, *ttemp;

void insertNodeBeforeVal (int val, int newVal) {
        ptr = first;
    while (ptr->data != val && ptr->next != NULL) {
                temp = ptr
                ptr = ptr->next;
            }

  if (ptr->data == val) {
                ttemp = (node*) malloc (sizeof (node));
                ttemp->data = newVal;
                ttemp->next = ptr;
      if (ptr == first)  first = ttemp;
      else               ptr->next = ttemp;
    }
  else printF ("%d. not found!", val);

}
```

Date 14/9/22

Expt. No. 11

Page No. 11

⇒ WAP to search a value in linked list.

```
typedef struct SinglyLinkedList {
                    int data;
            struct SinglyLinkedList *next;
} node;

node* first, *ptr;

int search (int val) {
        ptr = first;    int index = 0;
  while (ptr->data != val && ptr->next != NULL) {
                    ptr = ptr->next;  index++;
                }

  if (ptr->data == val) {
        printf ("Value %d found at index %d", val, index);
        return 1;
    }
else { printf ("Value %d not found!", val);
        return 0; }
}
```

Date 14/9/22

Expt. No. 12

Page No. 12

⇒ WAP to traverse a linked list.

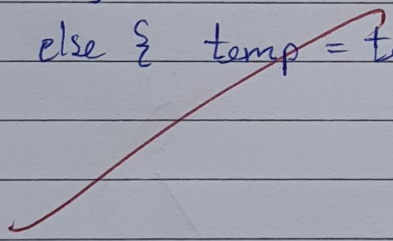
```
typedef struct SinglyLinkedList {
                int data;
        struct SinglyLinkedList *next;
} node;

node *first, *ptr;

void traverse() {
        if (first == NULL) {
                printf("Linked List is empty!");
                return;
        }
    ptr = first;
    int count = 1;
    while (ptr != NULL) {
            printf("Element %d = %d\n", count, ptr->data);
            ptr = ptr->next;
            count++;
        }
}
```

Date 19/10/22

Expt. No. 13 Page No. 13

⇒ WAP to remove duplicate element from linked list.

```
void deleteDuplicate() {
        temp = first;
    if (temp → next == NULL) {
        printf("There is only one node!");
    }
    else {
        while (temp → next != NULL) {
            if (temp → data = temp → next → data) {
                Ttemp = temp → next;
                temp → next = temp → next → next;
                Ttemp → next = NULL;
                free (Ttemp);
            }
            else { temp = temp → next; }
        }
    }
}
```



Date 19/10/22

Expt. No. 14

Page No. 14

⇒ WAP to find and display the middle element of linked list.

```
void displayMid(){
        int count = 0;
        temp = first;
        while (temp != NULL){
                count++;
                temp = temp -> next;
            }
        temp = first;
        for (int i = 0; i < count/2 ; i++){
                temp = temp -> next;
            }
        printf ("The middle value is %d \n", temp -> data);
}
```

Output :-

(15) ⇒ 5 repeated 3 times
3 repeated 3 times
2 repeated 3 times
4 repeated 2 times

Date 19/10/22

Expt. No. 15 Page No. 15

⇒ WAP to find duplicate element in array and its count in the array.

```
int isInArray(int arr[], int len, int key){
        for(int i=0; i<len; i++){
            if(arr[i]==key) return 1;
        }
        return 0;
    }

void printRepeatedCount(int arr[], int len){
    int alreadyCounted[len];
    int noOfAlreadyCounted = 0;
    for(int i=0; i<len; i++){
      if(isInArray(alreadyCounted, len, arr[i])) continue;
      alreadyCounted[noOfAlreadyCounted] = arr[i];
      noOfAlreadyCounted++;

      int count=1;
      for(int j=i+1; j<len; j++){
          if(arr[i]=arr[j]) count++;
      }
    if(count>1) printf("%d repeated %d times\n", arr[i], count);
    }
}

int main(){ int arr[]={5,3,2,3,5,2,7,1,5,4,3,2,4};
            printRepeatedCount(arr, 13);
            return 0; }
```

Date 19/10/22

Expt. No. 16

Page No. 16

⇒ WAP to show push & pop of stack using array datastructure.

```
#define max 5;

typedef struct Stack {
            int data[max];
            int top;
        } S;

int isFull (S *ptr) { return ptr->top == max-1; }

int isEmpty (S* ptr) { return ptr->top == -1; }

void push (S *ptr, int val) {
        if (isFull (ptr)) printf ("Stack Overflow!");
        else {
            (ptr -> top)++;
            ptr->data [ptr -> top] = val;
        }
}

int pop (S *ptr) { if (isEmpty(ptr)) printf ("Stack Underflow!");
        else {
            int val = ptr -> data [ptr -> top];
            (ptr -> top)--;
            return val;
        }
}
```

Teacher's Signature : _____

Date 19/10/22

Expt. No. 17

Page No. 17

⇒ WAP to show push & pop of stack using linked list data structure.

```
typedef struct Stack {
            int data;
            struct Stack * next;
    } Snode ;

Snode  *top , *temp;
top = NULL;

int isEmpty () { return  top == NULL; }

void push ( int val ){
        temp = (Snode*) malloc (sizeof(snode));
        temp -> next = top;
        temp -> data = val;
        top = temp;
    }

int pop (){
    if (isEmpty()) printf (" Stack Underflow ! \n");
    else {
            int val = top -> data;
            temp = top;
            top = top -> next;
            temp -> next = NULL;
            free (temp);
            return val;
    }
}
```

20/10/22

Teacher's Signature :

⇒ WAP to show enqueue & dequeue of queue using array data structure

```
#define max 5

typedef struct Queue {
            int data[max];
            int front ; int rear;
          } queue ;

int isFull (queue* q ) { return (q -> rear == max-1) ; }

int isEmpty (queue* q ) { return (q -> rear == -1 || q->front > q->rear); }

void enqueue (queue* q , int val) {
    if (isFull(q)) printf (" Queue is Full! \n ");
    else {
            if (q->front == -1 && q->rear == -1) {
                q->front = 0 ;   q->rear = 0;
              }
            else (q-> rear)++ ;
            q->data [q-> rear] = val;
            printf (" %d enqueued! \n", val);
    }
}

int dequeue (queue* q){
        if (isEmpty(q)) { printf (" Queue is already empty! \n");
      else {
            int dequeuedItem = q->data [q->front];
            (q->front)++ ;
```

Output 18 >>

```
10 enqueued!
20 enqueued!
30 enqueued!
40 enqueued!
50 enqueued!
Queue is Full!
10 dequeued
20 dequeued
30 dequeued
40 dequeued
50 dequeued
Queue is already empty!
```

indices →	-1	0	1	2	3	4	5
	f r	10 f r					
		10 f	20 r				
		10 f	20	30 r			
		10 f	20	30	40 r		
		10 f	20	30	40	50 r	
			20 f	30	40	50 r	
				30 f	40	50 r	
					40 f	50 r	
						50 f r	
							r f

Expt. No. 18 Page No. 19 Date

```
return dequeuedItem;
}

int main () {

    queue* myQ = (queue*) malloc (sizeof(queue));
    myQ->front = -1;
    myQ->rear = -1;

    enqueue (myQ, 10);
    enqueue (myQ, 20);
    enqueue (myQ, 30);
    enqueue (myQ, 40);
    enqueue (myQ, 50);

    enqueue (myQ, 60);  //overflow

    dequeue();
    dequeue();
    dequeue();
    dequeue();
    dequeue();

    dequeue();  //underflow

    return 0;
}
```

Date .....

Expt. No. 19

Page No. 20

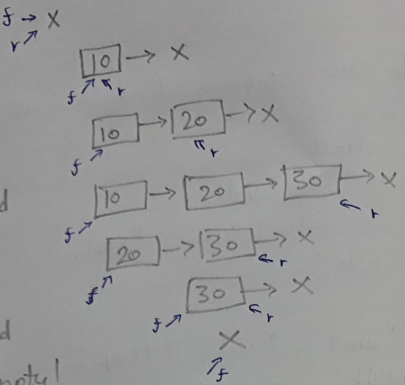
→ WAP to show enqueue & dequeue of queue using linked list.

```
typedef struct Queue {
            int val;
            struct Queue *next;
    } Qnode;

Qnode *front, *rear, *temp;
front = NULL;  rear = NULL;

int isEmpty () { return (front == NULL); }

void enqueue (int num) {
    temp = (Qnode*) malloc (sizeof (Qnode));
    temp → val = num;
    temp → next = NULL;
    if (rear != NULL) rear → next = temp;
    rear = temp;
    if (front == NULL) front = temp;
    printf ("%d enqueued \n", num);
}

int dequeue () {
    if (isEmpty()) printf ("Queue is Empty! \n");
    else {
        temp = front;  int val = temp → val;
        front = front → next;  if (front == NULL) rear = NULL;
        printf ("%d dequeued \n", temp → val);
        free (temp);  return val;
    }
}
```

Teacher's Signature : _____

Output 19 >>

10 enqueued
20 enqueued
30 enqueued
10 dequeued
20 dequeued
30 dequeued
Queue is Empty!



Expt. No. 19

```
int main(){
    enqueue(10);
    enqueue(20);
    enqueue(30);
    dequeue();
    dequeue();
    dequeue();
    dequeue(); //underflow;
  return 0;
}
```

→ WAP to show enqueue & dequeue of Circular queue using array.

```
#define max 5

int Cqueue[max];
int front = -1;  int rear = -1;

int isFull() { return ((rear == max-1 && front == 0) || (rear == front-1)); }

int isEmpty() { return (front == -1); }

void enqueue(int val) {
    if (isFull()) { printf("Queue is Full! \n"); }
    else {
        if (front == -1) front = 0;
        rear = (rear + 1) % max;
        Cqueue[rear] = val;
        printf("%d enqueued \n", val);
    }
}

int dequeue() {
    if (isEmpty()) { printf("Queue is Empty! \n"); }
    else {
        int valRemoved = Cqueue[front];
        if (front == rear) { front = -1; rear = -1; }
        else front = (front + 1) % max;
        printf("%d dequeued \n", valRemoved);
        return valRemoved;
    }
}
```

Teacher's Signature : _____

Output 20 »

10 enqueued

20 enqueued

30 enqueued

40 enqueued

50 enqueued

Queue is Full!

10 dequeued

20 dequeued

60 enqueued

70 enqueued

indices →	-1	0	1	2	3	4
	f r					
		10 f,r				
		10 f	20 r			
		10 f	20	30 r		
		10 f	20	30	40 r	
		10 f	20	30	40	50 r
			20 f	30	40	50 r
				30 f	40	50 r
		60 r		30 f	40	50
		60	70 r	30 f	40	50

Expt. No. 20

```
int main () {
    enqueue (10);
    enqueue (20);
    enqueue (30);
    enqueue (40);
    enqueue (50);
    enqueue (60); // overflow

    dequeue ();
    dequeue ();

    enqueue (60);
    enqueue (70);
    return 0;
}
```

Output 21 >> -23 48 2 -99 0

-99 -23 0 2 48

Date

Expt. No. 21

Page No. 24

⇒ WAP to perform Bubble Sort.

```
void displayArr (int arr[], int n){
        for (int i=0; i<n ; i++){
                printf ("%d ", arr[i]);
            }
        printf ("\n");
    }

void bubbleSort (int arr[], int n){
    For (int i=0 ; i<n ; i++) { //pass loop
        int swapped = 0;
        for (int j=1 ; j<n-i ; j++){ //comparison loop
            if (arr[j-1] > arr[j]){
                int temp = arr[j-1];
                arr[j-1] = arr[j];
                arr[j] = temp;
                swapped = 1;
            }
        }
        if (!swapped) return;
    }
}

int main (){
        int arr[5] = {-23, 48, 2, -99, 0};
        displayArr(arr, 5); //unsorted
        bubblesort (arr, 5);
        displayArr (arr, 5); //sorted
        return 0;
    }
```

Teacher's Signature :

Output 22 >> Unsorted arr: 4 2 -3 0 19
Sorted arr: -3 0 2 4 19

Expt. No. 22 Date

Page No. 25

⇒ WAP to perform insertion sort.

```
void displayArr(int arr[], int n){
        for(int i=0; i<n; i++){
            printf("%d ", arr[i]);
        }
        printf("\n");
    }

void insertionSort(int arr[], int n){
    for(int i=1; i<n; i++){
        for(int j=i; j>0; j--){
            if(arr[j] < arr[j-1]){
                int temp = arr[j];
                arr[j] = arr[j-1];
                arr[j-1] = temp;
            }
            else break;
        }
    }
}

int main(){ int arr[5] = {4, 2, -3, 0, 19};
        printf("Unsorted arr: ");
        displayArr(arr, 5);
        insertionSort(arr, 5);  printf("Sorted arr: ");
        displayArr(arr, 5);
        return 0;
}
```

Output 23 >> Unsorted Arr : 20 35 -15 7 1 -22
Sorted Arr : -22 -15 1 7 20 35

Expt. No. 23 Page No. 26

⇒ WAP to perform selection sort.

```
void display (int arr[], int n){
        for(int i=0 ; i<n ; i++){
            printf ("%d ", arr[i]);
        }
        printf ("\n");
    }

void selectionSort (int arr[], int n){
    for (int i=1 ; i<=n ; i++){ /* Pass loop */ int min=i-1;
        for (int j=i-1 ; i<=n-1 ; j++){
            if (arr[j] < arr[min]) min=j;
        }
        int temp = arr[i-1];
        arr[i-1] = arr[min];
        arr[min] = temp;
    }
}

int main(){
    int arr[6] = {20, 35, -15, 7, 1, -22};
    printf ("Unsorted Arr : ");
    display (arr, 6);
    selectionSort (arr, 6);
    printf ("Sorted Arr : ");
    display (arr, 6);
    return 0;
}
```

⇒ WAP to perform heap Sort.

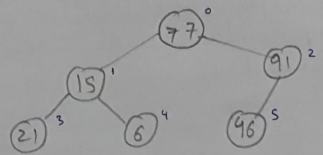
```
void swap (int arr[], int a, int b){
        int temp = arr[a];
        arr[a] = arr[b];
        arr[b] = temp;
    }

void display (int arr[], int n){
        for (int i=0; i<n; i++){ printf("%d ", arr[i]);}
        printf("\n");
    }

void maxHeapify (int arr[], int n, int i){
    int largest = i;
    int left = (2*i)+1;
    int right = left+1;

if (left<n && arr[left] > arr[largest]) largest = left;
if (right<n && arr[right] > arr[largest]) largest = right;

if (largest != i){
        swap (arr, largest, i);
        maxHeapify (arr, n, largest);
    }
}
```

[P.T.O]

Output 24 >> Unsorted arr: 77 15 91 21 6 46

Sorted arr: 6 15 21 46 77 91

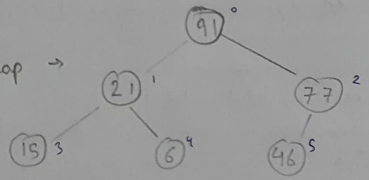
Heap →



77	15	91	21	6	46
0	1	2	3	4	5

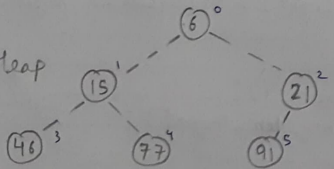
(Unsorted array)

Max Heap →



91	21	77	15	6	46
0	1	2	3	4	5

Min Heap



6	15	21	46	77	91
0	1	2	3	4	5

(Sorted array)

Date

Expt. No. 24

Page No. 28

```
void heapSort (int arr[ ], int n){
    for (int i = (n/2)-1 ; i>-1 ; i--) { // Build Max Heap
        maxHeapify (arr, n, i);
    }
    // now sort by "deletion"
    for (int i = n-1 ; i>=0 ; i--){
        swap (arr, 0, i);
        maxHeapify (arr, i, 0);
    }
}

int main ( ) {
    int arr[6] = { 77, 15, 91, 21, 6, 46};
    printf ("Unsorted arr: ");
    display (arr, 6);

    heapSort (arr, 6);

    printf ("Sorted arr: ");
    display (arr, 6);
    return 0;
}
```

Teacher's Signature :

Output 25 >> Enter size of array: 5
Enter elements:
1000
9
-82
0
27
The array is: 1000 9 -82 0 27
Maximum element is: 1000
Minimum element is: -82

Date

Expt. No. 25 Page No. 29

→ WAP to find minimum and maximum element in array.

```
#include <stdio.h>
int main() {

    int size;
    printf(" Enter size of array: ");
    scanf("%d", &size);
    int arr[size];

    printf("Enter elements: \n");
    for(int i=0; i<size; i++){
            scanf("%d", &arr[i]);
        }

    printf("The array is: ");
    int max = arr[0];
    int min = arr[0];

    for(int i=0; i<size; i++){
        if(arr[i] < min) min = arr[i];
        if(arr[i] > max) max = arr[i];

        printf("%d ", arr[i]);
    }
    printf("\nMaximum element is: %d\n", max);
    printf("Minimum element is: %d", min);
    return 0;
}
```

Output 26 >>

Array 1 : 0 2 4 6 8

Array 2 : 1 3 5 7 9

Concatenated Arr : 0 2 4 6 8 1 3 5 7 9

Date

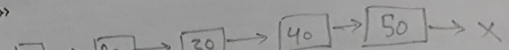
Expt. No. 26 Page No. 30

=> WAP to concatenate two arrays

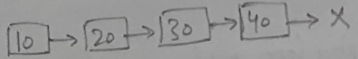
```
#include <stdio.h>

int display(int arr, int n){
        for(int i=0; i<n; i++){
                printf("%d ",arr[i]);
        }
        printf("\n");
}
int main(){
        int n1=5, n2=5;
        int arr1[n1] = {0,2,4,6,8}; // array 1
        int arr2[n2] = {1,3,5,7,9}; // array 2
        int n = n1+n2;
        int arrConcatenated[n];
        int index=0;
    for(int i=0; i<n1; i++){
                arrConcatenated[index] = arr1[i];
                index++;
        }
    for(int i=0; i<n2; i++){
                arrConcatenated[index] = arr2[i];
                index++;
        }
    printf("Array 1: "); display(arr1, n1);
    printf("Array 2: "); display(arr2, n2);
    printf("Concatenated Arr: "); display(arrConcatenated, n);
    return 0;
}
```

Teacher's Signature :

Output 27 »



deleteLast();



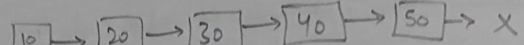
Date

Expt. No. 27 Page No. 31

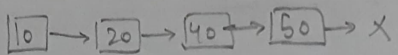
⇒ WAP to delete the last node of linked list.

```
typedef struct SinglyLL{
            int val;
            Struct SinglyLL *next;
    }node;

node *first, *temp, *ptr

void deleteLast(){
        if (first == NULL) return;
        ptr = first;
        while (ptr->next != NULL){
                temp = ptr;
                ptr = ptr->next;
            }
        if (ptr == first){
                first = first->next;
                ptr->next = NULL;
                free(ptr);
            }
        else { temp->next = NULL;
                ptr->next = NULL;
                free(ptr); }
    }
```

Output 28 >>



delete Val (30);



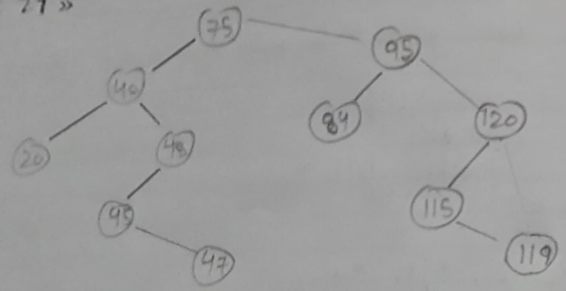
Date

Expt. No. 28 Page No. 32

⇒ WAP to delete a particular value from linked list.

```
typedef struct SinglyLL {
            int data;
            struct SinglyLL *next;
    } Node;

Node * first, *ptr, *temp;

void deleteVal (int num){ if (first==NULL) return;
        ptr = first;
        while (ptr != NULL && ptr->data!=num){
                temp = ptr;
                ptr = ptr->next;
            }
        if (ptr==NULL){ printf("Value not found!\n");
        else {
                if (ptr != first){
                        temp->next = ptr->next;
                        ptr->next = NULL;
                        free (ptr);
                    }
                else { first = first->NULL;
                        ptr->next = NULL;
                        free (ptr); }
            }
}
```

Output 29 »



preorder()	75	40	20	48	45	47	95	84	120	115	119
inorder()	20	40	45	47	48	75	84	120	115	119	
post order()	20	45	45	48	40	84	119	115	120	95	75

Date

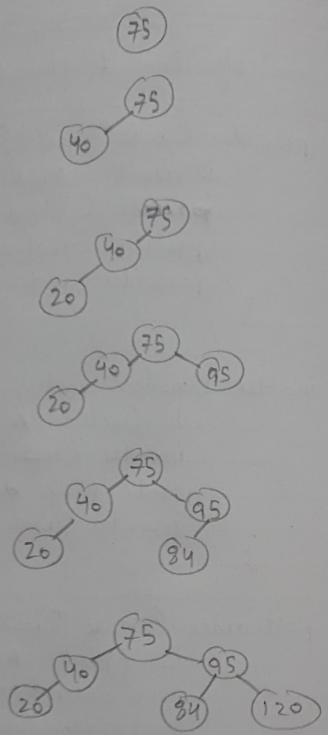
Expt. No. 29 Page No. 33

⇒ Write an algorithm to show traversal in tree

```
void preorder (node * root) {
        if (root == NULL) return;
        printf ("%d", root -> data);
        preorder (root -> left);
        preorder (root -> right);
    }

void inorder (node * root) {
        if (root == NULL) return;
        inorder (root -> left);
        printf ("%d", root -> data);
        inorder (root -> right);
    }

void post order (node * root) {
        if (root == NULL) return;
        post order (root -> left);
        postorder (root -> right);
        printf ("%d ", root -> data);
    }
```

75, 40, 20, 95, 84, 120,

Output 30 >> insert BST(75)
insert BST(40)
insert BST(20)
insert BST(95)
insert BST(84)
insert BST(120)



Date .....

Expt. No. 30 Page No. 84

⇒ Write an algorithm to perform insertion and search in BST.

```
void insertBST (Node * root, int val) {
        if (root == NULL) {
            root = (Node*) malloc (sizeof(Node));
            root -> data = val;
            root -> left = NULL;
            root -> right = NULL;
        }
    else if (root -> data > val) { insertBST (root -> left, val);
    else insertBST (root -> right, val);
}

Node* searchBST (Node * root, int key) { if (root == NULL) return 0;
        if (key == root -> data) return root;

        else if (key < root -> data) searchBST (root -> left, key);

        else searchBST (root -> right, key);

}
```