Output

```
Enter size of elements of array : 5
Enter the elements : 1
2
3
4
5
The sum is : 15
```

Date

Page No.

1. WAP in C to add ~~three~~ 'n' numbers using dynamic memory allocation :

```
#include <stdio.h>
int main()
{
    int n, *p, i, sum = 0
    printf("Enter the number of numbers you want to
            add :\t");
    scanf("%d", &n);
    p = (int*) malloc (n * sizeof(int));
    for(i=0; i<n; i++){
        scanf("%d", &p[i]);
        sum += p[i];
    printf("%d", sum);
    return 0;
}
```

Teacher's Signature : _____

Date .....

Expt. No. ..... Page No. .....

2. Write a program to find the greatest element of an array.

Source Code

```
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
void main() {
    int *a, n, i, max;
    printf("Enter the number of elements in the array:");
    scanf("%d", &n);
    a = (int *) malloc (n * sizeof(int));
    printf("Enter the elements :");
    for(i=0; i<n; i++)
        scanf("%d", &a[i]);
    max = a[0];
    for(i=0; i<n; i++){
        if(a[i] > max){
            max = a[i]
        }
    }
    printf("The greatest element in the array is %d",
            max);
    getch();
}
```

Teacher's Signature : _____

Expt. No. Date

Page No.

3. Write a program to perform Linear Search in an Array :

Source Code

```
#include <stdio.h>
#include <stdlib.h>
void main() {
        int *a, n, i, e, e=0;
        puts("Enter the size of elements in the array:");
        scanf("%d", &n);
        a = int *
```

Teacher's Signature : _____

Date .....

Expt. No. ..... Page No. .....

4 WAP to perform BINARY_SEARCH using dynamic memory allocation:

Source code :

```
# include <stdio.h>
# include <stdlib.h>
int main() {
    int *arr, size, i, num, f=0, l, j, c, mid, f1=0;
    printf("Enter size of array:");
    scanf("%d", &size);
    arr = (int *) malloc (size * sizeof(int));
    printf("Enter elements of the array:");
    for(i=0; i<size; i++){
        scanf("%d", &arr[i]);
    }
    printf("Enter number to be searched:");
    scanf("%d", &num);
    l = size-1;
    for (i=0; i<size-1; i++){
        for(j=0; j<(size-i-1); j++) {
            if (arr[j] > arr[j+1]) {
                c = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = c;
            }
        }
    }
```

Teacher's Signature : _____

Output

```
Enter size of array
5
Enter elements of array
1 7 9 8 2
Enter element to search
7
Element is in 2 position
```

Date .....

Expt. No. ..... Page No. .....

```
while (f<l) {
        mid = (f+l)/2;
        if (num > arr[mid]) {
                f= mid + 1;
        else if (num < arr[mid])
                l= mid - 1;
        else {
                printf ("Element is in %d position", mid);
                fl = 1;
                break;
        }
}

if (fl == 0)
        printf ("Number does not exist in list.");

return 0;
}
```

Teacher's Signature : _____

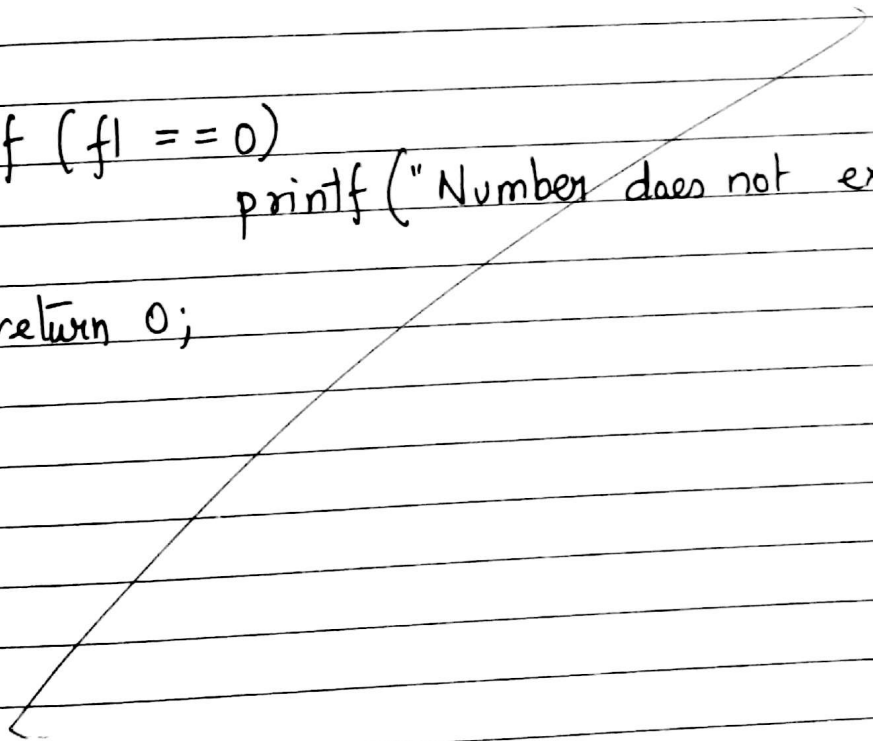
Date

Expt. No. Page No.

```
while (f<l) {
        mid = (f+l)/2;
        if (num > arr[mid]) {
                f= mid + 1;
        else if (num < arr[mid])
                l= mid - 1;
        else {
                printf ("Element is in %d position", mid);
                fl = 1;
                break;
        }
}

if (fl == 0)
        printf ("Number does not exist in list.");

return 0;
}
```



Teacher's Signature :

Expt. No. ..... Date .....

Page No. .....

5. WAP in C to perform BUBBLE SORT using dynamic memory allocation:

Source code

```
#include <stdio.h>
#include <stdlib.h>
int main() {
        int *arr, i, j, c=0;
        printf("Element array");
        scanf("%d" &size);
        arr = (int *) (malloc)(size * sizeof(int));
        for(i=0; i<size; i++)
                scanf("%d", &arr[i]);
        for(i=0; i<=(size-1); i++) {
                for(j=0; j<(size-1-i); j++){
                        if(arr[j] > arr[j+1]{
                                c = arr[j];
                                arr[j] = arr[j+1];
                                arr[j+1] = c;
                        }
                }
        }
        printf("Sorted array\n");
        for(i=0; i<size; i++)
                printf("%d", arr[i])
        return 0;
}
```

23/12/18

Teacher's Signature : _____

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

typedef struct nd {
        int data;
        struct nd * link;
} node;

node *start;

void display()
{
        node *p;
        if (start == NULL){
                printf("List is empty\n");
                exit(0);
        }
        p = start;
        while (p != NULL) {
                printf("%d\n", p -> data);
                p = p -> link;
        }
}
```

Teacher's Signature : _____

```
void addAtBeg() {
    node *temp;
    int val;
    printf("Enter the value");
    scanf("%d", &val);
    temp = (node*) malloc (sizeof(node));
    temp -> data = val;
    temp -> link = start;
    start = temp;

void addAtEnd() {
    node *p, *temp;
    int val;
    printf("Enter the value");
    scanf("%d", &val);
    temp = (node*) malloc (sizeof(node));
    temp -> data = val;
    p = start;
    while (p -> link != NULL) {
        p = p -> link;
        p -> link = temp;
        temp -> link = NULL;
    }

void addAtPos() {
    node *p, *temp;
    int i, val, pos;
    printf("Enter the val");
```

```
        scanf ("%d", &val);
        printf ("Enter the pos");
        scanf ("%d", &pos);
        temp = (node *) malloc (sizeof (node));
        temp -> data = val;
        p = start;
        for (i=1; i< pos-1 && p!= NULL ; i++) {
                p = p -> link;
                if (p == NULL)
                        p = p -> link; printf ("Not enough");
                else {
                        temp -> link = p -> link;
                        p -> link = temp;
                }
        }

void deteleAtBeg () {
        node *p;
        if (start == NULL) {
                print ("List is empty");
                exit (1);
        }
        p = start;
        start = start -> link;
        free (p);
}
```

Teacher's Signature : _____

Date .....

Expt. No. ..... Page No. .....

```
void deleteAtEnd() {
        node *p, *q;

        if (start == NULL) {
                printf("List is empty");
                exit(2);
        }

        p = start;
        while (p->link != NULL) {
                q = p;
                p = p -> link;
                q -> link = NULL;
                free(p);
        }

void deleteAtPos() {
        node *p, *q;
        int i, pos;

        printf("Enter the position");
        scanf("%d", &pos);
        if (start == NULL) {
                printf("List is empty");
                exit(3);
        }

        p = start;
```

Teacher's Signature : _____

```
        for (i=1 ; i<pos && p!= NULL ; i++) {
                q=p;
                p= p -> link ;
        }
        if (p == NULL)
                printf ("Less elements \n") ;
        else {
                q -> link = p -> link ;
                free (p) ;
        }
}
 // driver program

 void main ()
{
     int ch ;
     start = NULL ;
     start = (node *) malloc (sizeof (node)) ;
     printf ("1 -> Display \n");
     printf ("2 -> Add at end \n");
     printf ("3 -> Add at beginning\n");
     printf ("4 -> Add at particular position\n");
     printf ("5 -> Delete at end \n");
     printf ("6 -> Delete at beginning\n");
     printf ("7 -> Delete at particular position\n");
     printf ("Enter your choice ");
     scanf ("%d", &ch);
```

Date .....

Expt. No. .....

Page No. .....

```
switch(ch)
{
        case 1 : display();
                 break;

        case 2 : addAtEnd();
                 break;

        case 3 : addAtBeg();
                 break;

        case 4 : addAtPos();
                 break;

        case 5 : deleteAtEnd();
                 break;

        case 6 : deleteAtBeg();
                 break;

        case 7 : deleteAtPos();
                 break;

        default : printf("Choice is invalid");
}
getch();
}
```

23 | 3

Teacher's Signature : _____