Algorithm –

LinearSearch (a, n, temp) :- Here a is an array of size n and temp is a variable that represents a value to be searched.

1) Repeat step 2 for i=1 to i=n
2) if (temp = a[i]) then
   print "Element Found"
   Exit
   [end if]
   [end loop step 1]
3) if (i = n+1)
   print "Element not Found"
   [end if]
4) END

Output –

Enter size of Array 5

10 18 30 21 9

Enter the number you want to search
21

Element found at loc 4

a) Arrays

1. Program of linear search

```
#include <stdio.h>
void main()
{   int a[100], i, temp, n;
    printf("Enter size of Array");
    scanf("%d", &n);
    for(i=0; i<n; i++)
    scanf("%d", &n);
    printf("Enter the number you want to search");
    scanf("%d", &item);
    for(i=0; i<n; i++)
    {   if(temp == a[i]) {
        printf("Element found at loc %d", i+1);
        break;    }
    }
    if(i==n)
    printf("Element Not Found");
    getch();
}
```

Algorithm –

BinarySearch (a, n, item, flag) :- Here a is the array of size n and item is the variable which holds the value to be searched.

1) Set flag = False
2) Set start = 1 , end = n
3) mid := (start + end)/2
4) Repeat step 5 while (start ≤ end) do
5) if (item = a[mid]) then
   set flag = True and Exit
   else if (item < a[mid]) then
   set end = mid - 1
   set mid = (start + end)/2
   else
   set start = mid + 1
   mid = (start + end)/2
   [end if]
   end loop step 4
6) if (flag = True) then
   print "Element Found"
   else
   print "Element not found"
   end if
7) END

Output –

```
How many numbers 5
Enter Numbers
5 10 15 20 30
Enter the number you want to search 20
Element Found
```

Expt. No. _____ Date _____

Page No. 2

2. Program of Binary Search

```
#include <stdio.h>
void main()
{   int a[100], n, i, item, mid, start, end, flag=0;
    printf("How many numbers");
    scanf("%d", &n);
    printf("Enter numbers");
    for(i=0 ; i<n ; i++)
     scanf("%d", &a[i]);
     start=0;
     end = n-1;
     mid = (start + end)/2;
     while (start <= End ])
     {    if(item == a[mid]) {
           flag=1;
           exit(0);   }
          else if (item < a[mid]) {
           end = mid-1;
           mid = (start+end)/2; }
          else {
           start = mid+1;
           mid= (start + end)/2;  }
     }
     if (flag==1)
     printf("Element Found");
     else
     printf("Element Not Found");
      getch();
}
```

Teacher's Signature _____

Date

Expt. No. Page No. 13

3. Insertion into Array at any position

```
#include <stdio.h>
void main()
{       int a[100], i, n, item, loc;
        printf("Enter size of array");
        scanf("%d", &n);
        printf("Enter Numbers");
        for(i=0; i<n; i++)
        scanf("%d", &a[i]);
        printf("Enter Number to insert");
        scanf("%d", &item);
        printf("Enter location");
        scanf("%d", &loc);
        for(i=n-1; i>=loc-1; i--)
          a[i+1] = a[i];
          a[loc-1] = item;
          n++;
        printf("After Insertion");
          for(i=0; i<n; i++)
          printf("%2d", a[i]);
          getch();
}
```

Teacher's Signature

Algorithm –

DeleteIntoArray (a, n, loc) :- Here a is an array of size n and loc is the location from where we want to delete the element.

1) Repeat step 2 for i = loc to i < n
2) Set a[i] = a[i+1]
   end loop step 1
3) Set n = n-1
4) END

Output –

```
Enter size of Array 5
Enter Numbers
10  20  30  40  50
Enter the location 4
After Deletion
10  20  30  50
```

Date

Expt. No. _____ Page No. 4

4. Deletion into Array using location

```
#include <stdio.h>
void main()
{       int a[100], i, n, loc;
        printf("Enter size of Array");
        scanf("%d", &n);
        printf("Enter Numbers");
        for(i=0; i<n; i++)
        scanf("%d", &a[i]);
        printf("Enter the location");
        scanf("%d", &loc);
        for(i=loc-1; i<n-1; i++)
        a[i] = a[i+1];
        n--;
        printf("After Deletion");
        for(i=0; i<n; i++)
        printf("%3d", a[i]);
        getch();
}
```

Teacher's Signature _____

Algorithm-

DeleteInto Array (a, n, item, loc) :- Here a is an array of size n and loc is the location of the number item to be deleted.

1) Repeat step 2 for i=1 to i=n
2) if (a[i] = item) then
   Set loc = i
   [end if]
   end loop Step 1
3) Repeat step 4 for i= loc to i<n
4) Set a[i] = a[i+1]
   end loop step 4
   Set n= n-1
5) END

Output-

Enter size of array 5
Enter Numbers
10 20 8 11 4
Enter number to be deleted 11
After deletion
10 20 8 4

Expt. No. _____ Date _____

Page No. 5

5. Deletion into varray using element

```
#include <stdio.h>
void main()
{       int va[100], i, n, item, loc;
        printf("Enter size of varray");
        scanf("%d", &n);
        printf("Enter Numbers");
        for(i=0 ; i<n ; i++)
        scanf("%d", &va[i]);
        printf("Enter number to be deleted");
        scanf("%d", & item);
        for(i=0 ; i<n ; i++){
         if(va[i] == item)
         loc = i ;                    }
         for(i= loc ; i<n-1 ; i++)
           va[i] = va[i+1];
           n--;
         printf("After Deletion");
         for(i=0 ; i<n ; i++)
         printf("%3d", va[i]);
         getch();
}
```

Teacher's Signature _____

Algorithm –

BubbleSort (a, temp, n) :- Here a is an array of size n and temp is a temporary variable for swapping.

1) Repeat Step 2 for i=1 to i<n
2) Repeat Step 3 for j=1 to j<n-i
3) if (a[j] > a[j+1]) then

Set temp = a[j]

Set a[j] = a[j+1]

Set a[j+1] = temp

end if
end loop step 2
end loop step 1
4) END

Output –

Enter size of Array 6
Enter numbers 12 6 11 18 2 0
After Sorting
0 2 6 11 12 18

6. Bubble Sorting Program

```
#include <stdio.h>
int main()
{
    int a[100], i, j, temp;
    int n;
    printf("Enter size of Array");
    scanf("%d", &n);
    printf("Enter numbers");
    for(i=0; i<n; i++)
     scanf("%d", &a[i]);
    for(i=0; i<n-1; i++)
    {
        for(j=0; j<n-1-i; j++)
        {
            if(a[j] > a[j+1])
            {
                temp = a[j];
                a[j] = a[j+1];
                a[j+1] = temp;
            }
        }
    }
    printf("After sorting \n");
    for(i=0; i<n; i++)
     printf("%3d", a[i]);
     return 0;
}
```

Algorithm –

SelectionSort (a, temp, n) :- Here a is an array of size n and temp is the temporary variable for swapping.

1) Repeat step 2,3,5 for i=1 to i = n-1
2) Set min = a[i]
3) Repeat step 4 for j= i+1 to j = n
4) if (min > a[j]) then
   set min = a[j]
   set K = j
   end if
   end loop step 3
5) if (min != a[i]) then
   temp = a[i]
   a[i] = min
   a[K] = temp
   end if
   end loop step 1
6) END

Output –

```
Enter size of Array 6
Enter Numbers  50  41  60  32  18  90
The Sorted Array is
18  32  41  50  60  90
```

Date

Expt. No.

Page No. 7

7. Selection Sorting Program

```
#include <stdio.h>
void main()
{
        int a[100], i, j, k, temp;
        printf("Enter size of Array");
        scanf("%d", &n);
        printf("Enter Numbers");
        for(i=0 ; i<n ; i++)
        scanf("%d", &a[i]);
        for(i=0 ; i<n-1 ; i++)
        {
            min = a[i];
            for(j=i+1 ; j<n ; j++)
            {
                if(min > a[j])
                {
                    min = a[j];
                    k= j;
                }
            }
            if(min != a[i])
            {
                temp = a[i];
                a[i] = min;
                a[k] = temp;
            }
        }
        printf("The sorted Array is \n");
        for(i=0 ; i<n ; i++)
        printf("%d", a[i]);
        getch();
}
```

Teacher's Signature

Algorithm –

InsertionSort (a, n) :- Here a is an array of size n.

1) Repeat step 2 & 3 for $i = 2$ to $i = n$

   a) Set target = a[i]

   b) Set j = i-1

2) Repeat while target ≤ a[j] and J ≥ 1

   a) Set a[j+1] = a[j]

   b) Set j = j-1

   end loop step 2

3) Set a[j+1] = target

   end loop step 1

4) END

Output –

```
Enter size of Array 5
Enter the numbers
22  78  96  41  2
After Sorting
2  22  41  78  96
```

Date _____

Expt. No. _____ Page No. 8

8. Insertion Sort Program

```
#include <stdio.h>
void main()
{   int a[100], n, i, j, target;
    printf("Enter size of Array");
    scanf("%d", &n);
    printf("Enter the numbers");
    for(i=0; i<n; i++)
     scanf("%d", &a[i]);
    for(i=1; i<n; i++)
    {       target = a[i];
            j = i-1;
        while(target < a[j] && j>=0)
        {     a[j+1] = a[j];
              j--;
        }
        a[j+1] = target;
    }
    printf("After Sorting\n");
     for(i=0; i<n; i++)
       printf("%3d", a[i]);
       getch();
}
```

Teacher's Signature _____

Algorithm -

Merge_Sort (A, Beg, End) :- Here a is an array of size n. Here Beg stores the starting position of the first element and end stores the position of last most element.

1) if (Beg < End) then
   a) mid = (start + end)/2
   b) Merge_Sort (A, Beg, Mid)
   c) Merge_Sort (A, Mid+1, End)
   d) Merge (A, Beg, Mid, End)
   end if
2) EXIT

Merge (A, Beg, mid, end)

1) Set I = Beg & J = Mid+1 and K = Beg
2) Repeat step 3 while (I ≤ Mid And J ≤ END)
3) If (A[I] < A[J]) then
   Set [K] = A[I]
   Set I = I+1
   Set K = K+1
   else
   Set C[K] = A[J]
   Set J = J+1
   Set K = K+1
   [end if]
   [end loop step 2]
4) Repeat while (I ≤ Mid)
   a) Set C[K] = A[I]
   b) Set I = I+1
   c) Set K = K+1

9. Merge Sorting Program

```
#include <stdio.h>
void main()
{       int a[20], n;
        printf("Enter size of array");
        scanf("%d", &n);
        printf("Enter elements");
        for(i=0; i<n; i++)
        scanf("%d", &a[i]);
        mergesort(0, n-1);
        printf("Sorted Array is");
        for(i=0; i<n; i++)
        printf("%3d", a[i]);
}
mergesort(int Beg, int End)
{       int mid;
        if(Beg < End)
        {       mid = (Big + End)/2;
                mergesort(Big, mid);
                mergesort(mid+1, End);
                merge(Big, mid, End);
        }
}
merge(int Big, int mid, int End)
{       int i, J, K, C[20];
        i = Big;
        J = mid+1;
        K = Big
        while(i<=mid && J<=End)
```

[end loop step 4]

5) Repeat while (I ≤ End)

a) Set C[K] = A[J]

b) Set J= J+1

c) Set K= K+1

[end loop step 5]

6) Repeat for I= Beg to End

Set A[I] = C[I]

[end loop step 6]

7) Return

Output –

Enter size of array 6
Enter elements 10 5 8 1 4 21
Sorted Array is 1 4 5 8 10 21

```
{       if ( a[i] < a[J])
        {       c[K] = a[i];
                i++;
                K++;
        }
        else {
                c[K] = a[J];
                J++;
                K++;
        }
}
while ( i <= mid)
{       c[K] = a[i];
        i++;
        K++;
}
while ( J <= End)
{       c[K] = a[J];
        J++;
        K++;
}
for(i=0; i<n; i++)
        a[i] = c[i];
}
```

Algorithm –

Quick_Sort (a, first, last) :- Here a is an array. First is the starting index of array. last is the end index of array.

1) Start
2) if(first < last) then
   Set pivot = first
   Set i = first
   Set j = last
3) while (i < j) do
   a) while a[i] <= a[Pivot] And i < last do
   b) Set i = i+1
   end while loop a)
4) while a[j] > a[Pivot]
   a) Set j = j-1
   end while loop step 5
5) if(i < j) then
   swap (a[i], a[j])
   end if Step5
   end if step2
6) swap (a[pivot], a[j])
7) quicksort (a, first, j-1)
8) quicksort(a, j+1, last)
9) EXIT

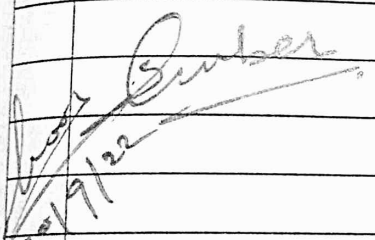
10. Quick Sort Program

```
#include <stdio.h>
void main()
{       int a[40], n, i;
        printf("Enter size of Array");
        scanf("%d", &n);
        printf("Enter elements");
        for(i=0; i<n; i++)
        scanf("%d", &a[i]);
        quicksort(a, 0, n-1);
        printf("Sorted elements are");
        for(i=0; i<n; i++)
          printf("%d", a[i]);
          return 0;
}
void quicksort(int a[], int First, int Last)
{       int i, J, pivot, temp;
        if(First < Last)
        {   pivot = First;
            i = First;
            J = Last;
        }
        while(i<J)
        {   while(a[i] <= a[pivot])
            i++;
            while(a[J] > a[pivot])
            J--;
            if(i<J)
            {   temp= a[i];
```

Output-

```
Enter size of Array 5
Enter Elements  10  9  5  11  4
Sorted Elements are  4  5  9  10  11
```

```
            a[i] = a[J];
            a[J] = temp;
        }
    }
    temp = a[pivot];
    a[pivot] = a[J];
    a[J] = temp;
    quicksort (a, 0, J-1);
    quicksort (a, J+1, Last);
}
```



Algorithm –

SearchInLL (FLAG, ITEM, PTR, START, INFO) :- Here PTR is a pointer variable and ITEM is the number we want to search. This algo will find the number in the linked list.

1) if (START = NULL) then
   print " List is Empty"
   return.
   end if
2) set PTR = START
3) Repeat step 4 while (PTR ≠ NULL)
4) if (ITEM = INFO [PTR]) then
   set FLAG = 1
   Break
   else
   PTR = LINK [PTR]
   end loop step 2
5) if (FLAG = 1) then
   print " Element Found"
   else
   print " Element Not Found"
   end if
6) Exit

Expt. No. _____ Date _____

Page No. 13

b) Linked Lists

1. Function to search in linklist

```
struct node
{   int info;
    struct node * link;
} * start = NULL;

void search()
{   int item;
    int flag=0;
    struct node * ptr;
    printf(" Enter the number you want to search");
    scanf("%d", & item);
    ptr = start;
    if ( start == NULL)
    {  printf(" Empty list");
       return;
    }
    else {
          while ( ptr != NULL) {
              if ( item == ptr -> info){
                  flag = 1;  break;} ptr = ptr->link;
          }
         if ( flag == 1)
          printf(" Element Found"); else
          printf("Element Not Found");  }
}
```

Teacher's Signature _____

Algorithm –

Insert_Beg (ITEM, INFO, START, NEW) :- Here ITEM is the number we want to add in the linkedlist and START points to the first node in memory.

1) if (AVAIL = NULL) then
   print " Overflow"
   EXIT
   end if
2) Set NEW = AVAIL
3) Set AVAIL = LINK[AVAIL]
4) Set INFO[NEW] = ITEM
5) Set LINK[NEW] = START
6) Set START = NEW
7) EXIT

2. Function to insert a node at the begining of linked list

```
struct node
{    int info;
     struct node * link;
} * start = NULL;

void insert_beg ()
{    struct node * new;
     int item;
     new = (struct node *) malloc (sizeof(struct node));
     printf (" Enter the number ");
     scanf ("%d", &item);
     if (start == NULL)
     {    start = new;
          new -> link = NULL;
          new -> info = item;    }
     else {
          new -> link = start;
          start = new;
          new -> info = item;    }
}
```

Algorithm –

Insert_End (INFO, START, ITEM, NEW, PTR) :- Here ITEM is the number we want to insert in the linked list at the end. PTR is a pointer variable that will point the first node.

1) if (AVAIL = NULL) then
   print "Overflow"
   Exit
   end if
2) Set NEW = AVAIL
   Set AVAIL = LINK [AVAIL]
   Set PTR = START
3) Repeat step 4 while (LINK [PTR] ≠ NULL)
4) Set PTR = LINK [PTR]
5) Set LINK [PTR] = NEW
6) Set LINK [NEW] = NULL
7) Exit

Date

Expt. No. _____ Page No. 15

3. Function to insert a node at the end of linklist.

```
struct node {
    int info;
    struct node * link;
} * start = NULL;

void insert_end ()
{   struct node * new; * ptr;
    int item;
    new = (struct node *) malloc ( sizeof ( struct node));
    new -> info = item;
    ptr = start;
    if ( start == NULL) {
        start = new;
        new -> link = NULL; }
    else {
        while ( ptr -> link != NULL)
        {   ptr = ptr -> link;
        }
        ptr -> link = new;
        new -> link = NULL;
    }
}
```

Teacher's Signature _____

Algorithm –

Insert_In_Between (INFO, NUMBER, ITEM, START, PTR) :- Here ITEM is the number we want to add in the list after the number NUMBER.

```
1)  if ( AVAIL = NULL ) then
        print " Overflow"
        EXIT
2)  Set     NEW = AVAIL
    Set     AVAIL = LINK[AVAIL]
    Set     INFO[NEW] = ITEM
    Set     PTR = START
3)  Repeat step 4 while ( PTR ≠ NULL)
4)          if(INFO[PTR] = NUMBER) then
                Set FLAG1 = 1
                Exit
            else
                Set PTR = LINK[PTR]
            [end if]
            end loop step3
5)  if ( FLAG1 = 1) then
    Set     PTR = START
            Repeat step 6 while ( INFO[PTR] ≠ NUMBER)
6)              Set PTR = LINK[PTR]
                end loop
                end if
7)          LINK[NEW] = LINK[PTR]
8)          LINK[PTR] = NEW
9)  if (FLAG1 ≠ 1) then
        print " Element is not in the list"     [end if]
10) END
```

Date

Expt. No. _____ Page No. 16

4. Function to insert a node between two nodes by using element.

```
struct node {
        int info;
        struct node * link;
} * start = NULL;
void insert_between()
{   struct node * new, * ptr;
    int item, number;
    printf ("Enter the number you want to insert");
    scanf("%d", &number);
    printf (" Enter the number after which you want
            to insert");
    scanf("%d", &item);
    ptr = start;
    while (ptr != NULL) {
        if( item == ptr → info) {
            flag=1; break; }
        else { ptr = ptr → link; }
        if ( flag == 1) {
            ptr = start;
            while( ptr → info != item) {
                ptr = ptr → link; }
    new → link = ptr → link;
    ptr → link = new;       }
        else
            printf (" Element is not in the list");
}
```

Teacher's Signature _____

Algorithm –

Delete_Beg(START, ITEM, INFO) Here ITEM is a variable that stores the deleted number from the link list.

1) if (START = NULL) then
   print " Underflow"
   EXIT
2) Set TEMP = START
3) Set ITEM = INFO[START]
4) Set START = LINK[START]
5) Set LINK[TEMP] = AVAIL
6) Set AVAIL = TEMP
7) EXIT

Date

Expt. No. _____ Page No. 17

5. Function to delete first node in linked list

```
struct node {
  int info;
  struct node * link;
} * start = NULL;
void Delete_Beg()
{  struct node *ptr;
   int item;
   if ( start == NULL)
   {  printf(" List is Empty");
      return;
   }
   else {
      ptr = start;
      item = start -> info;
      start = start -> link;
      free(ptr);
      printf(" Deleted Number is %d", item);
   }
}
```

Teacher's Signature _____

Algorithm–

Del-From-End (INFO, START, LINK, TEMP, PREV, PTR) %- Here PREV, START, LINK, PTR are the pointer variables. This algo will delete the last node from the linkedlist.

1) if (START = NULL) then
   print " Underflow "
   EXIT
   [end if]
2) if (LINK[START] = NULL) then
   a) Set TEMP = START
   b) Set START = NULL
      go to step 10
      [end if]
3) Set PREV = NULL AND PTR = START
4) Repeat step 5 & 6 while (LINK[PTR] ≠ NULL)
5) Set PREV = PTR
6) Set PTR = LINK[PTR]
   [end loop step 5]
7) Set TEMP = PTR
8) Set LINK[PREV] = NULL
9) Set ITEM = INFO[PTR]
10) Set LINK[TEMP] = AVAIL
11) AVAIL = TEMP
12) EXIT

qp 9

Date

Expt. No. _____ Page No. 12

6. Function to delete a node from the end of list.

```
struct node {
    int info ;
    struct node * link ;
} * start = NULL ;
void Del-End ()
{   struct node * ptr , * temp , * prev ;
    int item ;
    if ( start == NULL) {
     printf (" list is empty");
      return ;        }
    else if ( Start → link == NULL) {
            temp = start ;
            start = NULL ;
             free ( temp) ;
        }
        else
        {       prev = NULL ;
                ptr = start ;
                while ( ptr → link != NULL)
                {       prev = ptr ;
                        ptr = ptr → link ;
                }
                temp = ptr ;
                prev → link = NULL ;
                item = ptr → info ;
                free ( ptr) ;
        }
}
```

Teacher's Signature _____

Algorithm–

Del given (NUMBER, ITEM, INFO, PREV, PTR, START) Here PREV, PTR, START are the pointer variables and NUMBER is the element we want to delete from the list.

1) if (START = NULL) then
   print " Underflow"
   EXIT

2) if (LINK [START] = NULL AND INFO[START] = NUMBER) then
   a) Set TEMP = START
   b) Set ITEM = INFO [START]
   c) Set START = NULL
   d) Set LINK [TEMP] = AVAIL
   e) Set AVAIL = TEMP AND EXIT
   end if

3) Set PTR = START

4) Set PREV = NULL

5) Repeat step a and b while (LINK[PTR] != NULL AND INFO [PTR] ! = Item)
   a) PREV = PTR
   b) PTR = LINK [PTR]
   end loop step 5

6) Set temp = ptr

7) Set item = INFO [PTR]

8) Set LINK [PREV] = LINK [PTR]

9) Set LINK [PTR] = AVAIL

10) Set AVAIL = PTR

11) EXIT.

Date

Expt. No. Page No. 19

7. Function to delete a given element in linkedlist.

```
struct node {
    int info;
    struct node * link;
} * start = NULL;
void Del_between()
{   struct node * temp, * prev, * ptr;
    int item, number;
    printf(" Enter the number you want to delete");
    scanf("%d", &number);
    if( start == NULL) {
      printf(" List is empty");
      return;   }
    else if( start -> link == NULL) {
        if( start -> info == number ) {
            item = start -> info ;
            START = NULL ;
            free (start);      }
    } else {
      ptr = start ;   prev = NULL;
      while( ptr -> link != NULL && ptr -> info != number)
      {     prev = ptr ;
            ptr = ptr -> link ;
      }
        temp = ptr ;
        item = ptr -> info;
        prev -> link = ptr -> link ;
        free (ptr);      }
}
```

Teacher's Signature

Algorithm –

Count_Nodes (COUNT, PTR, START) :- Here START and PTR is a pointer variable and COUNT denotes no. of nodes in linked list.

1) PTR = START
2) Repeat steps 3 & 4 while (PTR != NULL)
3) Set count = count + 1
4) Set PTR = LINK [PTR]
   end loop step 2
5) EXIT

Date

Expt. No. _____ Page No. 20

8. Function to count total number of nodes in linked list

```
struct node {
    int info;
    struct node * link;
} * start = NULL;
void count_nodes ()
{   struct node *ptr;
    int count = 0;
    ptr = start;
    while ( ptr != NULL) {
        count ++;
        ptr = ptr → link;
    }
    printf (" Nodes in the list is :%d", count);
}
```

Teacher's Signature _____

Algorithm –

Display (PTR , INFO , START) :- Here START and PTR are the pointer variables. This algo will print the linkedlist.

1) if (START = NULL) then
   print " list is Empty"
   EXIT
2) PTR = START
3) Repeat step 4 and 5 while (PTR != NULL)
4) print " INFO[PTR]
5) PTR = LINK[PTR]
6) END

Date

Expt. No.

Page No. 21

9. Function to display linklist

```
struct node {
    int info;
    struct node * link;
} * start = NULL;
void display ()
{       struct node * ptr;
        if ( start == NULL) {
          printf(" List is empty ");
          return;}
        else
        {   ptr= start;
            while (ptr != NULL)
            {   printf ("%5d" , ptr -> info);
                ptr = ptr -> link;
            }
        }
}
```

Teacher's Signature

Output -

```
Select your choice
1. Insert Node at Begining
2. Insert Node at End
3. Insert Node in Between
4. To delete First node
5. To delete Last node
6. To search in linkedlist
7. To delete a element
8. To count number of nodes
9. To print linklist

9
List is Empty
1
Enter the number 10
2
Enter the number 20
2
Enter the number 30
9
10      20      30
5
Deleted item is 30
8
Nodes in the list is 2
9
10    20
7
Enter the Number you want to search   20
Element Found
```

Date

Expt. No.

Page No. 22

10. Various operations on linkedlist

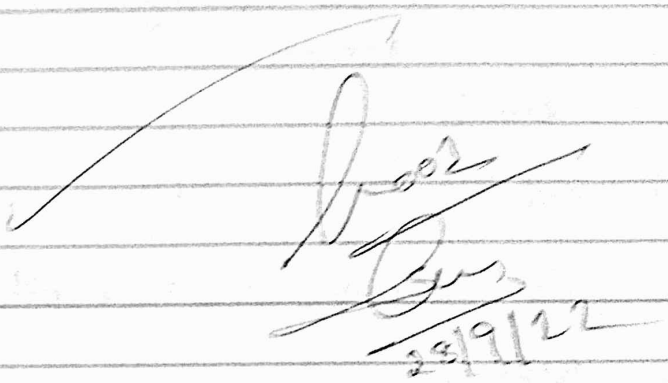
```
struct node {
        int info;
        struct node * link;
} * start = NULL;
void main() {
    int choice;
    printf(" Select your choice ");
    printf(" 1. \n Insert Node at Begning");
    printf(" 2. \n Insert Node at the end");
    printf(" 3. \n Insert Node in Between ");
    printf(" 4. \n To delete First node");
    printf(" 5. \n To delete last node ");
    printf(" 6. \n To delete a element ");
    printf(" 7. \n To search in linkedlist");
    printf(" 8. \n To count number of nodes ");
    printf(" 9. \n To print linklist");
    while (1)
     {  scanf("%d", & choice);
        switch (choice)
        {   case 1:  insert-beg();  break;
            case 2:  insert-end();  break;
            case 3:  insert-between();  break;
            case 4:  Delete-Beg();  break;
            case 5:  Del-End();  break;
            case 6:  Del-between();  break;
            case 7:  search();  break;
            case 8:  count-nodes();  break;
```

Teacher's Signature

Date

Expt. No. Page No. 23

```
        case 3 : display() ; break ;
        default :
            exit(0) ;
    }
}
    getch() ;
}
```



29/9/22

Algorithm –

PUSH(STACK, TOP, MAX, ITEM):- Here TOP points the topmost element in STACK and ITEM is the element we want to insert into STACK.

1) if (TOP = MAX) then
   print "Overflow"
   Return
   end if
2) Set TOP = TOP + 1
3) Set STACK[TOP] = ITEM
4) Return

c) Stacks Using Arrays

1. Function for insertion (push) onto stack.

```
void push()
{   int item;
    printf("Enter item to be insert: ");
    scanf("%d", &item);
    if (top == max-1)
    {   printf("Overflow");
        return;
    }
    else
    {   top++;
        stack[top] = item;
    }
}
```

Algorithm –

POP(STACK, TOP, ITEM) – This algo deletes top element of stack and assign it to the variable ITEM.

1) if (top = 0) then
   print " Underflow"
   Return
   end if
2) Set ITEM = STACK [TOP]
3) Set TOP = TOP - 1
4) Return

Date

Expt. No. _____ Page No. 25

2. Function for deletion (pop) onto the stack.

```
void pop()
{   int item;
    if (top == -1)
    {   printf (" Underflow");
        return;
    }
    else
    {   item = stack [top];
        top--;
        printf (" Deleted number is %d\n", item);
    }
}
```

Algorithm –

DISPLAY (STACK, TOP) :- This algo will print the elements of STACK.

1) if (top == 0) then
   print " Empty Stack"
   Return
   end if
2) Repeat step 3 for i=TOP to 1
3) Print STACK[i]
4) Return

Date

Expt. No.

Page No. 26

3. Function to display stack elements.

```
void display()
{       if (top == -1)
            printf(" Empty Stack\n");
        else
        { for (int i= top ; i>=0 ; i--)
            printf("%5d", stack[i]);
            printf ("\n");
        }
}
```

Teacher's Signature

Output –

```
1. Push
2. Pop
3. Display
Enter choice : 1
Enter item to be insert : 10
Enter choice : 2
Deleted number is 10
Enter choice : 3
Empty stack
```

Date

Expt. No. _____ Page No. 27

4. Basic operations on stack using arrays

```
#define MAX 20
int stack[max], n;
int top = -1;
void main()
{   int choice;
    while (1)
    {  printf("1. Push \n");
       printf("2. Pop \n");
       printf("3. Display \n");
       scanf("%d", &choice);
       switch (choice)
       {   case 1:
               push();   break;
           case 2:
               pop();    break;
           case 3:
               display(); break;
           default :
               exit();
       }
    }
}
```

Teacher's Signature _____

Algorithm –

PUSH(AVAIL, ITEM) :- This algo will insert ITEM onto the STACK

1) if (AVAIL = NULL) then
   print "overflow"
   return
   end if
2) Set NEW = AVAIL
3) Set AVAIL = LINK [AVAIL]
4) Set INFO [NEW] = ITEM
5) Set LINK [NEW] = TOP
6) Set TOP = NEW
7) EXIT

Date

Expt. No. Page No. 26

Stacks Using Linkedlists

1. Function for insertion (push) onto the stack.

```
void push()
{   struct node * new;
    int item;
    new = (struct node *) malloc (sizeof( struct node));
    printf (" Enter item you want to push :");
    scanf("%d", & item);
    if (top == NULL)
    {   top = new;
        new → link = NULL;
        new → info = item;
    }
    else
    {   new → link = top;
        top = new;
        new → info = item;
    }
}
```

Teacher's Signature

Algorithm –

POP (TOP, STACK) :- This algo deletes the topmost element of the STACK

1) if (TOP = NULL) then
   print " Underflow "
   Return
   end if
2) Set TEMP = TOP
3) Set ITEM = INFO [TOP]
4) Set TOP = LINK [TOP]
5) LINK [TEMP] = AVAIL
6) AVAIL = TEMP
7) EXIT

Date

Expt. No.

Page No. 29

2. Function for deletion (pop) onto the stack.

```
void pop()
{   struct node * temp;
    int item;
    if (top == NULL)
    {   printf(" Underflow");
        return;
    }
    else
    {   temp = top;
        item = top -> info;
        top = top -> link;
    }
    printf(" Popped item is %d \n", item);
    free( temp);
}
```

Teacher's Signature

Algorithm–

Display (PTR, STACK, TOP) :- This algo will print the stack elements

1) if (TOP = NULL) then
   print " Empty Stack"
   Return
   end if
2) Set PTR = TOP
3) Repeat Step 4 while PTR ≠ NULL
4) Print INFO[PTR]
   Set PTR = LINK [PTR]
   end loop Step 3
5) END

Date

Expt. No. _____ Page No. 30

3. Function to print stack elements.

```
void display ()
{   struct node *ptr;
    if (top == NULL)
    {   printf(" Empty List");
        return;
    }
    else
    {   ptr = top;
        while (ptr != NULL)
        {   printf("%4d", ptr→info);
            ptr = ptr→link;
        }
    }
}
```

Teacher's Signature _____

Output –

```
1. Push
2. Pop
3. Display
Enter choice : 1
Enter item you want to push : 10
Enter choice : 1
Enter item you want to push : 20
Enter choice : 3
20  10
Enter choice : 2
Deleted item is 20
```

Date

Expt. No. _____ Page No. 31

4. Basic operations on stack using linkedlist.

```
void main()
{   struct node
    {   int info;
        struct node *link;
    } *top = NULL;
    int choice;
    while (1)
    {   printf("1. Push\n");
        printf("2. Pop\n");
        printf("3. Display\n");
        scanf("%d", &choice);
        switch (choice)
        {   case 1:
                push(); break;
            case 2:
                pop(); break;
            case 3:
                display(); break;
            default:
                exit();
        }
    }
}
```

HOD

28/9/22

Teacher's Signature _____

Output –

```
Enter infix :  3*2+6
Postfix : 32*6+
Value of expression: 12
Want to continue (y/n) : y
Enter infix : 2^3+6/2+8%5
Postfix: 23^62/+85%+
Value of expression: 14
Want to continue (y/n) : n
```

Date

Expt. No. _____ Page No. 32

5. Program for conversion of infix to postfix and evaluation of postfix

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#define Blank ' '
#define Tab '\t'
#define MAX 50
long int pop();
long int eval_post();
char infix[MAX], postfix[MAX];
long int stack[MAX];
int top;

int prec(char symbol)
{   switch(symbol)
    {   case '(': return 0;
        case '+':
        case '-': return 1;
        case '*':
        case '/':
        case '%': return 2;
        case '^': return 3;
    }
}
void push(long int symbol)
{   if(top > MAX)
    {   printf("Stack Overflow\n");
        return;
    }
```

Teacher's Signature _____

Algorithm of conversion of infix to postfix –

CONVERSION(Q, STACK) :- Here Q is an arithmetic expression written in infix notation.

1) Push "#" onto STACK and add "#" at the end of Q

2) Scan Q from left to right and repeat step 3 to 6 for each element of Q until the stack is empty.

3) if an operand is encountered, add it to P

4) if a left paranthesis is encountered, push it onto stack.

5) if an operator ⊗ is encountered, then:

   a) Repeatedly pop from stack and add to P each operator (on TOP of stack) which has the same precedence as or higher precedence than ⊗

   b) Add ⊗ to STACK

   [end if]

6) If a right paranthesis is encountered, then:

   a) Repeatedly Pop from STACK and add to P each operator (on the top of STACK) until a left paranthesis is encountered.

   b) Remove the "#" [DO not add it to P].

   [end if]

7) Exit

Date

Expt. No. Page No. 33

```
        else {
                top ++ ;
                stack [ top ] = symbol ;
             }
}
void infix_to_postfix ()
{   int i , p=0 , type , precedence , len ;
    char next ;
    stack [ top ] = '#' ;
    len = strlen ( infix ) ;
    infix [len] = '#' ;
    for ( i=0 ; infix [i] != '#' ; i++)
    {   switch ( infix [i] )
        {   case '(' :    push (infix[i]); break ;
            case ')' :
            while ((next = pop()) != '(')
             postfix [p++] = next ; break ;
             case '+' :
             case '-' :
             case '*' :
             case '/' :
             case '.' :
             case '^' :
            precendence = prec ( infix[i]) ;
            while ( stack[ top ] != '#' && precedence <=
                                    prec (stack[top]))
            postfix [p++] = pop ();
            push ( infix [i]);      break;
            default:    postfix [p++]= infix [i] ;
        }
}
```

Teacher's Signature

Algorithm of evaluation of postfix –

Evalpost (P, STACK) :- Here P is an arithmetic expression written in postfix.

1) Add "#" at the end of P
2) Scan P from left to right and repeat Step 3 and 4 for each element of P until "#" is encountered.
3) If an operand is encountered, push it on STACK.
4) If an operator ⊗ is encountered, then:
   a) Remove two top elements of STACK. where A is the top element and B is the next to top element.
   b) Evaluate B ⊗ A
   c) Place the result of b) back on STACK
   [end if]
   [end loop Step 2]
5) Set VALUE equal to the top element of STACK.
6) EXIT


```
        }
        while (stack[top] != '#')
         postfix[p++] = pop();
         postfix[p] = '\0';
}
  long int pop()
  {     if (top == -1) {
        printf(" Stack Underflow \n");
          return;     }
        else
          return (stack[top--]);
  }
long int eval-post()
{       long int temp, result, len;
        int i,
        char va, b;
        len = strlen(postfix);
        postfix[len] = '#';
        for(i=0 ; postfix[i] != '#' ; i++)
        {     if(postfix[i]<='9' && postfix[i]>='0')
                push (postfix[i] -48);
              else {
                va= pop();
                b = pop();
                switch (postfix[i])
                {   case '+' :   temp= b+va ; break;
                    case '-' :   temp = b-va ; break;
                    case '*' :   temp= b*va; break;
                    case '/' :   temp = b/va; break;
```

Teacher's Signature _____

```
                case '%' : temp = b % a ; break;
                case '^' : temp = pow(a,b); break;
            }
            push (temp);
        }
    }
    result = pop ();
    return result;
}
void main ()
{   long int value;
    char choice = 'y';
    while (choice == 'y')
    {   top = 0;
        printf ("Enter Infix : ");
        fflush (stdin);
        gets (infix);
        infix-to-postfix ();
        printf ("Postfix : %s\n", postfix);
        value = eval-post ();
        printf ("Value of expression: %d\n", value);
        printf ("Want to continue (y/n): ");
        scanf ("%c", &choice);
    }
}
```

Algorithm –

LQINSERT (Queue, Max, front, rear, item) :- Here 'queue' is an array representing queue with maximum 'Max' elements. Item is the value to be inserted.

1) if (rear == Max) then
   print "Overflow"
   Return
   end if

2) if (front = 0 and rear = 0) then
   set front = 1
   set rear = 1
   else
   set rear = rear + 1
   [end if-else]

3) Set Queue [rear] = item

4) EXIT

Expt. No. _____ Date _____ Page No. 26

d) Linear Queue Using Array

1. Program of insertion into linear Queue

```
#include <stdio.h>
#include <conio.h>
void insert()
{   int item;
    if ( rear == Max-1)
    {   printf("Overflow");
        return;
    }
    if ( front == -1)
        front = rear = 0;
    else
     rear++;
    printf("Enter item:");
    scanf("%d", &item);
     Queue[rear] = item;
}
```

Teacher's Signature _____

Algorithm –

LQDELETE(Queue, Max, front, rear, item) :- Here 'Queue' is an array representing queue with maximum 'Max' elements.

1) if (front = 0 and rear = 0) then
   print "Underflow"
   Return
   [end if]
2) Set item = Queue [front]
3) if (front = rear) then
   Set front = rear = 0
   else
   Set front = front + 1
   [end if]
4) print Item
5) EXIT

Date

Expt. No. _____ Page No. 27

2. Program of deletion into linear Queue

```
# include <stdio.h>
# include <conio.h>
void delete ()
{   int item;
    if ( front == -1)
    {  printf(" Underflow");
       return;
    }
    item = Queue [ front];
    if ( front == rear)
       front = rear = -1;
    else
       front ++;
    printf(" Deleted item is %d", item);
}
```

Teacher's Signature _____

Algorithm –

DISPLAY (Queue, Max, front, rear) :- Here 'queue' is an array representing queue with maximum 'Max' elements.

1) if (front = 0)
   print " Queue is Empty "
   Return
   [end if]

2) Repeat step 3 for i = front to rear

3) print Queue [i]

4) EXIT

Date

Expt. No.

Page No. 32

3. Program to display linear Queue

```
#include <stdio.h>
#include <conio.h>
void display()
{     int i;
      if ( front == -1)
      { printf (" Queue is Empty");
        return;
      }
      for ( i = front; i <= rear ; i++)
            printf ("%5d", Queue [i]);
}
```

Teacher's Signature

Output –

```
1. Insert
2. Delete
3. Display
Enter choice : 1
Enter item to be inserted: 100
1. Insert
2. Delete
3. Display
Enter choice: 3
100
```

Date

Expt. No. _____ Page No. 39

4. Various operations on linear Queue

```
int Queue[MAX], front = -1, rear = -1;
# include <stdio.h>
# include <conio.h>
void main()
{   int choice;
    while (1)
    {   printf (" 1. Insert \n");
        printf (" 2. Delete \n");
        printf (" 3. Display \n");
        printf (" Enter choice : ");
        scanf (" %d", & choice");
        switch ( choice)
        {   case 1:  insert ();    break;
            case 2:  delete ();    break;
            case 3:  display ();   break;
            default: exit ();
        }
    }
}
```

Teacher's Signature _____

Algorithm –

CQINSERT(Queue, Max, front, rear, item) :- This algo inserts item at rear of circular queue

1) if(front = rear+1 OR front =1 AND rear = Max) then
   print " Overflow"
   Return

   [end if]

2) if (front =0 AND rear =0) then
   set front = 1
   set rear = 1
   else if (rear = Max) then
   set rear = 1

   else
   set rear = rear +1

   [end if-else]

3) Set Queue [rear] = item

4) EXIT

Date

Expt. No.

Page No. 40

Circular Queue Using Array

1. Program of insertion into circular Queue

```
# include < stdio.h>
# include < conio.h>
void insert ()
{       int item;
    if((front == 0 && rear == Max-1)|| front == rear +1)
    {    printf (" Overflow");
         return;
    }
    printf(" Enter item to insert ");
     scanf("%d", & item);
     if ( front == -1)
         front = rear = 0 ;
     else
     {       if ( rear == Max-1)
                 rear = 0 ;
             else
                 rear ++;
      }
        Queue [ rear] = item;
}
```

Teacher's Signature

Algorithm–

CQDELETE (Queue, Max, front, rear) :- This algo will delete element from front of circular Queue

1) if (front = 0 AND rear = 0) then
   print " Underflow"
   Return
   [end if]

2) Set item = Queue [front]

3) if (front = rear) then
   Set front = 0
   Set rear = 0
   else
   if (front = Max) then
   set front = 1
   else
   set front = front + 1
   [end if-else]
   [end if-else]

4) EXIT

Date

Expt. No.

Page No. 41

2. Program of deletion into circular Queue

```
# include <stdio.h>
# include <conio.h>
void delete()
{       int item;
        if ( front = = -1)
        {   printf (" Overflow");
            return;
        }
        item = Queue [ front];
        printf (" Deleted item is %d", item);
        if ( front = = rear)
            front = rear = -1;
        else
        {   if ( front == Max-1)
                front = 0
            else
                front ++;
        }
}
```

Teacher's Signature

Algorithm -

DISPLAY (Queue, Max, front, rear) :- This algo will display elements of circular Queue.

1) if (front = 0) then
   print " Empty Queue "
   Return
   [end if]

2) if (front < rear) then

3) Repeat step a for i = front to rear
   a) print " Queue [i]

4) else
   Repeat step b for i=1 to rear
   b) print Queue [i]

5) Repeat step c for i= front to Max
   c) print Queue [i]
   [end if-else]

6) EXIT

Date

Expt. No. _____ Page No. 42

3. Program to display circular Queue

```
#include <stdio.h>
#include <conio.h>
void display()
{     int i;
      if( front == -1)
      {  printf(" Queue is empty");
         return;
      }
      if( front <= rear)
      {  for(i=0; i < front; i++)
            printf(" ");
         for(i= front; i <= rear; i++)
            printf("%d", Queue[i]);
      }
      else
      {  for( i=0; i <= rear; i++)
            printf("%d", Queue[i]);
         for( i= rear+1; i < front; i++)
            printf(" ");
         for( i= front; i < Max; i++)
            printf("%d", Queue[i]);
      }
}
```

Teacher's Signature _____

Output –

1. Insert
2. Delete
3. Display

Enter choice: 1

Enter item to insert: 100

1. Insert
2. Delete
3. Display

Enter choice: 2

Deleted item is 20

Expt. No. _____ Date _____

Page No. 43

4. Various operations on circular Queue

```
#include <stdio.h>
#include <conio.h>
#define Max 20
int Queue[Max], front = -1, rear = -1;
void main()
{   int choice;
    while (1)
    {   printf(" 1. Insert \n");
        printf(" 2. Delete \n");
        printf(" 3. Display \n");
        printf(" Enter choice: ");
        scanf(" %d", &choice);
        switch (choice)
        { case 1:  insert();   break;
          case 2:  delete();   break;
          case 3:  display();  break;
          default: exit();
        }
    }
}
```

Teacher's Signature _____

Algorithm –

AQINSERT(INFO, LINK, FRONT, REAR, ITEM) :- This algo will insert ITEM into Queue.

1) if (AVAIL = NULL) then
   print " Overflow"
   Return.
   [end if]

2) Set NEW = AVAIL
3) Set AVAIL = LINK[AVAIL]
4) Set INFO[NEW] = ITEM
5) Set LINK [NEW] = NULL
6) if (FRONT = NULL) then
   a) Set FRONT = NEW
   b) Set REAR = NEW
   else
   Set LINK [REAR] = NEW
   Set REAR = NEW
   [end if]
7) EXIT

Expt. No. _____ Date _____ Page No. 44

Queue Using linkedlist

1. Function to insert in Queue

```
# include < stdio.h>
# include < conio.h>
void insert()
{   struct node * new;
    int item;
    new = (struct node *) malloc (sizeof (struct node));
    if ( new == NULL)
    { printf (" Overflow \n");
      return ;
    }
    printf ("Enter item: ");
    scanf("%d", & item);
    new → info = item ;
    new → link = NULL ;
    if ( front == NULL)
     front = rear = new ;
    else
    {  rear → link = new ;
       rear = new ;
    }
}
```

Teacher's Signature _____

Algorithm –

DELQueue (FRONT, REAR, LINK, INFO, ITEM) :- This algo will delete the item which is at FRONT location.

1) if (FRONT = NULL) then
   print "Underflow"
   Return
   [end if]

2) temp = FRONT

3) if (LINK[FRONT] = NULL) then
   a) set FRONT = NULL
   b) set REAR = NULL
   else
   set FRONT = LINK [FRONT]
   [end if]

4) ITEM = INFO [temp]

5) Set LINK[temp] = Avail

6) Set AVAIL = temp

7) EXIT

Expt. No. _____ Date _____ Page No. 45

2. Function to delete in Queue

```
#include <stdio.h>
#include <conio.h>
void delete()
{   struct node * temp;
    int item;
    if ( front == NULL)
    {  printf (" Underflow\n");
       return;
    }
    temp = front;
    item = front -> info;
    if ( front -> link == NULL)
       front = rear = NULL
    else
    front = front -> link;
    printf (" Deleted item is %d\n", item);
    free ( temp);
}
```

Teacher's Signature _____

Algorithm –

DISPLAY (PTR, FRONT, REAR, LINK) :- This algo will display or print the Queue elements.

1) if (FRONT = NULL) then
   print " List is Empty"
   Return
   [end if]
2) Set PTR = FRONT
3) Repeat step 4 while PTR ≠ NULL
4) Print INFO [PTR]
   a) set PTR = LINK [PTR]
   end loop step 3
5) EXIT

Expt. No. _____ Date _____ Page No. 46

3. Function to display queue elements

```
#include <stdio.h>
#include <conio.h>
void display()
{   struct node *ptr;
    if ( front == NULL)
    {  printf (" Empty Queue \n");
       return;
    }
    ptr = front;
    while ( ptr != NULL)
    {  printf ("%4d", ptr -> info);
       ptr = ptr -> link;
    } printf ("\n");
}
```

Teacher's Signature _____

Output –

```
1. Insert
2. Delete
3. Display
Enter choice: 1
Enter item: 100
1. Insert
2. Delete
3. Display
Enter choice: 2
Deleted item is 100
1. Insert
2. Delete
3. Display
Enter choice: 3
Empty Queue
```


Date

Expt. No.

Page No. 47

4. Various functions on queue using linkedlist

```
#include <stdio.h>
#include <conio.h>
struct node
{   int info;
    struct node * link;
} * front = NULL, * rear = NULL;

void main()
{   int choice;
    while (1)
    {   printf("1. Insert\n");
        printf("2. Delete\n");
        printf("3. Display\n");
        printf("Enter choice: ");
        scanf("%d", &choice);
        switch (choice)
        {
            case 1: insert(); break;
            case 2: delete(); break;
            case 3: display(); break;
            default: exit();
        }
    }
}
```

Teacher's Signature