Algorithm :

Step 1 : Start

Step 2 : Declare variables i, j, n

Step 3 : Initialise name[10][50], temp[25]

Step 4 : Read n for number of string

Step 5 : Set i=0 and Read Strings one by one until i<n

Step 6 : Set j=0 Increment j until j<=n-i

Step 7 : If (strcmp (name[j], name[j+1]) > 0 then

copy(temp, name[j]), (name[j], name[j+1]),

(name[j+1], temp)

Step 8 : Print the string after Sorting

Step 9 : Stop.

Output :

Input number of strings : 3

Input 3 String :

Shifa

Hadiya

Mirsha

The string after Sorting is :

Hadiya

Mirsha

Shifa.

Date ..... Expt. No ..... 1 ..... Page No ..... 2 .....

Aim : To Sort a list of strings

Program :

```
#Include <stdio.h>
#Include < string.h>
#Include < conio.h>
  void main( )
{
  Chas name[10][50], temp[25];
  Int n, i, j ;
  Clrscr ( );
Printf (" Input number of strings :");
Scanf ("%d", &n );
Printf ("%d Input %d String :", n);
  for (i=0 ; i <n ; i++)
Scanf ("%s"; name[i]);
  for (i = 1 ; i <=n ; i++)
 for (j=0 ; j <=n-i ; j++)
If (Strcmp (name[j]; name.[j+1]) >0)
{
  Strcpy (temp, name[j] );
  Strcpy ( name[j], name [j+1]);
  Strcpy (name[j+1], temp );
}
```

Teacher's Signature: .....

```
printf("The strings after sorting is :\n");
    for(i=0; i<n; i++)
printf("%s \n", name[i]);
    getch();
```

Teacher's Signature: .....

Algorithm :

Step 1 : Start

Step 2 : Declare the variable i, len and str [100]

Step 3 : get str for enter a string

Step 4 : Set i = 0 Increment i until i<100

Step 5 : Find length of string using str len

Step 6 : If (*ptr == '\0') Increment ptr
len = i decrement ptr

Step 7 : Set i = len decrement i until i>0

Step 8 : Print the Reverse of string

Step 9 : Stop.

Output :

Enter a String : Malappuram

Reversed String : maruppalam

Date ..... Expt. No ..... 2 ..... Page No ..... 4 .....

Aim : To reverse a string using pointers

Program :

```
#include <stdio.h>
#include <conio.h>
void main()
{
char str[100], *ptr;
int i, len;
clrscr();
printf ("enter a string");
gets (str);
  ptr = str;
for (i=0; i<100; i++)
{
  if (*ptr == '\0')
    break;
    ptr++;
}
len = i;
ptr--;
printf("Reversed string:");
for (i=len; i>0; i--)
  printf("%c", *ptr--);
  getch();
}
```

Teacher's Signature: .....

Algorithm :

Step 1 : Start

Step 2 : Declare R, S, k, L, MAX, INDEX

Step 3 : Read P for enter text of pattern matching

Step 4 : Read T for enter the string

Step 5 : set R = Str len (P)

S = Str len (T)

MAX = S - R

Step 6 : Set L = 0 Increment L until L < R

Step 7 : If (P[L] != T[K+L])

Step 8 : If (L == 0) then Index = k, else k = k+1

If k > MAX then INDEX = -1

Step 9 : print the values of P and T

Step 10 : If INDEX != -1 then print Index of P in T

Step 11 : Stop

Output :

Enter the text to for pattern matching : Puram

enter the string : Malappuram

P = Puram , T = Malappuram

Index of P in T is 6

Date ..... Expt. No ..... 3 ..... Page No ..... 5 .....

Aim : pattern Matching

Algorithm

Program :

```
#include <stdio.h>
#include <conio.h>
  void main()
{
 char p[80], T[80];
 int R,S,k,L,MAX,INDEX;
 clrscr();
 printf("\n enter the text for pattern matching:");
 scanf("%s",P);
 printf("\n enter the string");
 Scanf("%s",T);
 R = strlen(P);
 S = strlen(T);
 k=0
MAX = S-R ;
while (k<=MAX
{
 for(L=0;L<R;L++)
 if(P[L]!=T[K+L])
  break;
 if(L==R)
{
```

Teacher's Signature: .....

```
INDEX = k;
  break;
else
  k = k+1;
}
 if (k > MAX)
 INDEX = -1;
 Printf ("P = %s", P);
 printf ("\n\n T = %s", T);
 If (INDEX != -1)
printf ("\n\n Index of P in T is %d", INDEX +1);
 else
printf ("\n\n P does not exist in T");
 getch ();
}
```

Teacher's Signature: .....

Algorithm :

Step1 : Start

Step2 : Declare variable i, j, k, m, n, a_len, b_len

Step 3 : Initialise array a[20], b[20], c[40]

Step 4 : Read m for enter no:of elements in 1st array

Step 5 : Set i = 0 Increment i until i<m

Step 6 : Read a[i] for enter the elements

Step 7 : Read n for enter no:of elements in 2nd array

Step 8 : Set i = 0 increment i until i<n

Step 9 : Read b[i] for enter the elements

Step 10 : Set c[i] = a[i]

Step 11 : Read a[i]. print array one

Step 12 : Read b[i] print array two

Step 13 : set i = 0 Increment until i<m+n

Step 14 : print array after appending

Step 15 : Stop.

Output :

Enter number of elements in first array : 4

Enter the elements : 3 1 6 9

Enter number of elements in second array : 3

Date ..... Expt. No .....4..... Page No .....7.....

Aim : To append two arrays

Program :

```
#include <stdio.h>
#include<string.h>
#include <conio.h>
void main()
{
int a[20], b[20], c[40], a_len, b_len, i, j, k, m, n;
clrscr();
Printf ("enter number of elements in first array");
Scanf(" %d", &m);
Printf ("enter the elements:");
 for (i=0; i<m ; i++)
{
 Scanf ("%d", &a[i]);
}
Printf for Printf ("enter number of element in second array");
Scanf ("% d", &n );
Printf ("enter the elements:");
 for (i=0 ; i<n ; i++)
{
 Scanf ("% d", & b[i]);
}
for (i=0; i<m ; i++)
```

Teacher's Signature: .....

Enter the elements : 7 4 2

Array one : 3 1 6 9

Array two : 7 4 2

Array after appending : 3 1 6 9 7 4 2

```
{
  c[i] = a[i];
}
printf ("Array one");
for (i=0; i<m; i++)
{  printf ("%d", a[i]);
}
printf ("\n");
printf ("Array two");
for (i=0; i<n; i++)
{
  printf ("%d", b[i]);
}
printf ("\n");
printf ("Array after appending:");
 for (i=0; i<m+n; i++)
{
  printf ("%d", c[i]);
 }
}
```

Teacher's Signature: .....

Algorithm :

Step 1 : Start

Step 2 : Declare variable c, last, middle, n, item

Step 3 : Initialise fast = 0 , array[100]

Step 4 : Read n for enter no:of elements

step 5 : Set c = 0 Increment c until c < n

Step 6 : Read item for enter value to find

Step 7 : last = n - 1, while (first <= last)

Step 8 : middle = (first + last)/2

Step 9 : If (array[middle] < item) then

first = middle + 1

else if (array[middle == item)

Step 10 : print the location found

Step 11 : Stop

Output :

Enter number of elements : 5

Enter 5 Integers : 23 67 89 101 120

Enter value to find : 67

67 found at location 2.

Date .....

Aim :

Program

#incl

#incl

void

{

int

clrs

printf

Scan

print

for

Scan

prin

Sc

la

wh

{

m

If

f

e

}

Date ..... Expt. No .....5..... Page No .....9.....

Aim : To Search an element in one-dimensional array using Iterative Binary search

Program :

```
#include <stdio.h>
#include <conio.h>
void main()
{
int c, first=0, last, middle, n, item, array[100];
clrscr();
printf("enter number of elements \n");
scanf("%d",&n);
printf("enter %d integers \n",n);
for(c=0; c<n; c++)
scanf("%d",&array[c]);
printf("enter value to find \n");
Scanf("%d",&item);
last = n-1;
while(first <= last)
{
middle =(first + last)/2;
If (array[middle] < item)
first = middle +1;
else if (array[middle] == item)
{
```

Teacher's Signature: .....

```
printf ("%d found at location %d \n", item,
middle +1);
  break;
}
  else
last = middle -1;
}
 if (first > last)
printf ("not found ! %d isn't present in the
      list \n", item);
}
```

Teacher's Signature: .....

Algorithm :

Step 1 : Start

Step 2 : Declare variables i, j, m, n, c, k and a[20][20]

Step 3 : Read n, m for enter number of Rows & columns

step 4 : Repeat step 5 to step 8 for i=0 to i = m-1

Step 5 : Repeat steps to 8 for j=0 to j = n-1

Step 6 : Read elements to a[i][j]

Step 7 : If a[i][j] != 0 Increment c otherwise go to step 8

Step 8 : Increment k

[end of If structure]

[end of loop]

Step 9 : If c < k goto step 10 otherwise goto Step 13

Step 10 : Repeat. Step 11 to 12 for i=0 to i = m-1

Step 11 : Repeat Step 12 for j=0 to j = n-1

Step 12 : If a[i][j] != 0 print i\t j\t a[i][j]

[end of Inner loop]

[end of outer loop]

Step 13 : print not a space matrix

Step 14 : stop

Date ..... Expt. No .....6..... Page No .....11.....

Aim : Read a Sparse Matrix and display its triplet representation using array.

Program :

```
#Include <stdio.h>
#Include <conio.h>
 Void Main( )
{
 Int i, j, m, n, c =0, k=0;
 Int a[20][20];
 printf ("enter no:of rows and columns :");
 Scanf ("%d %d", &m, &n);
 printf ("enter Sparse matrix \n");
  for (i=0; i<m ; i++)
 for (j=0; j<n ; j++)
{
 Scanf ("%d", &a[i][j]);
 If (a[i][j]!=0)
  c++;
 else
  k++;
}
 If (c<k)
{
  Print f ("Row \t col\b value \n");
```

Teacher's Signature: .....

Output :

Enter no:of rows and colums : 3 3

Enter Sparse Matrix :

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 2 & 5 & 3 \end{bmatrix}$$

Raw	col	value
3	3	4
1	2	1
2	0	2
2	1	5
2	2	3

```
printf("%d \t %d \t %d \n", m, n, c);
  for (i=0; i<m ; i++)
{
  for (j=0; j<n ; j++)
{
  if (a[i][j] != 0)
printf ("%d \t %d \t %d \n", i, j, a[i][j]);
  }
}
}
else
  printf ("not a sparse Matrix \n");
}
```

Teacher's Signature: .....

Algorithm :

Step 1 : start

Step 2 : Globally declare Struct node *link with int data and Struct node *ptr, *tmp

Step 3 : Declare int variable n, i, item

Step 4 : Read n for enter number of nodes

Step 5 : set i=0 Increment i until i<n

Step 6 : tmp = (struct node *)malloc(Size of struct node)

Step 7 : Read item for enter the data

Step 8 : tmp → data = item

tmp → link = start

Start = tmp

ptr = Start

Step 9 : print the linked list with n nodes

step 10 : Stop.

Output :

Enter the number of nodes : 4

Enter the Data : 4 23 1 90

The linked list with 4 node is : 90 → 1 → 23 → 4 →

Date .....

Aim

Prog

#in

#in

St

{

Date ..... Expt. No .....7..... Page No .....13.....

Aim : Create a singly linked list of n nodes and display it

Program :

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
struct node
{
  int data ;
struct data *link ;
 }
*start = NULL , *ptr , *tmp ;
void main()
{
  int n , i, item ;
clrscr() ;
printf ("\n enter the no:of nodes:" );
scanf ("%d", &n );
for (i=0 ; i<n ; i++)
{
  tmp = (struct node *) malloc (size of (struct
    node));
printf ("\n enter the data :" );
scanf ("%d", & item);
```

Teacher's Signature: .....

```
tmp → data = item ;
tmp → link = start ;
start = tmp ;
}
ptr = start ;
printf ("In the linked list with %d nodes is:",n);
for (i=0 ; i<n ; i++)
{
printf ("%d →", ptr → data);
ptr = ptr → link ;
}
getch();
}
```

Teacher's Signature: .....

Algorithm :

Step 1 : Start
Step 2 : Declare variables
Step 3 : Create a node new node.
Step 4 : print the list of nodes *head
Step 5 : Set while (head)
Step 6 : print the header value.
set head = head → next.
Step 7 : Insert front nodes *head , Integer values.
Step 8 : Set Struct node * new node = NULL
Create a new node.
Step 9 : If new node = NULL then print call to Interst element out of memory.
Step 10 : Set new-node → val = value
new-node → next = *head.
*head = new_node.
Step 11 : Delete a node *head , Integer values.
Step 12 : Check If (head == NULL) then returns.
If (head → val == val) then true.
Step 13 : Set tmp = *head
*head = (*head) → next.
Step 14 : while (tmp → next and tmp → next → val not equal to val. then temp = tmp → next
Step 15 : else print the Integers not found in the list.
Step 16 : Declare the main functions Integer variables., i, val
count = 0.

Date .....

Aim

Prog

#In

#I

S

{

Date ..... Expt. No .....8..... Page No .....15.....

Aim : Delete a node from singly linked list.

Program :

```
#Include <stdio.h>
#Include <stdlib.h>
 Struct node
{
 int val ;
Struct node *next ;
 };
void print_list (struct node *head)
{
  while (head)
{
   Printf ("%d→", head → val);
   head = head → next ;
 }
Printf ("\n");
 }
void Insert_front (struct node * *head, int value)
{
 Struct node + new_node = NULL;
 new-node = (struct node *) malloc (size of (struct node));
  If (new_node = = NULL)
{
```

Teacher's Signature: .....

Step 17 : Create a linked list and nodes.

Step 18 : Read count for enter no: of elements.

Step 19 : set i=0 , i<count, increment i

Step 20 : Read val to enter i.th element.

Step 21 : print the initial list.

Step 22 : Read val for enter a value to delete.

Step 23 : Delete a node, header & value.

Step 24 : print the list After Deletion.

Step 25 : Stop.

Output :

Enter number of elements : 5

Enter elements : 1 2 3 4 5

Enter a value to delete : 3

List after Deletion : 1 → 2 → ~~3~~ → 5

```
printf ("call to interst element out of memory");
}
new_node → val = value ;
new_node → next = *head ;
*head = new_node ;
}
void delet_node (struct node * *head , int val)
{
struct node *tmp = NULL ;
struct node *tmp1 = NULL;
if (head = = NULL || *head = = NULL) return ;
if (head) → val = = val)
{
tmp = *head
*head = (*head) → next ;
free (tmp);
return ;
}
tmp = *head ;
while (tmp → next && tmp → next → val != val)
tmp = tmp → next ;
if (tmp → next)
{
tmp = tmp → next
tmp → next = temp1 → next ;
```

```
free (tmp1);
}
 else
 {
 printf ("%d not found in the list \n", val);
 }
 void main ()
{  int count =0, i, val ;
  Struct node *head =NULL;
 printf ("enter number of elements:");
 scanf ("%d", &count);
 for (i=0 ; i< count ; i++)
 {
 Printf ("Enter %d th element :", i);
 Scanf ("%d", &val);
Insert_front (&head, val);
 }
Print f ("Initial list :");
Printf ("enter a value to delete from the list:");
 Scanf ("%d", &val);
delete_node (&head, val);
printf ("list after deletion :");
 Print_list (head);
}
```

Teacher's Signature: .....

Algorithm :

Step 1 : start
Step 2 : define a structure node with data pointer to next node and left, right node
Step 3 : set *start = NULL
Step 4 : Declare variable ch, item
Step 5 : print the list 1. add
2. Display forward
3. Display Backward
4. exit
step 6 : Read ch to enter ur choice.
step 7 : start switch statement
case 1: Read item to enter items
set temp → data = item
temp → Left = NULL
temp → right = NULL
If (start == NULL), start = temp
case 2 : If start = NULL, print no element display
Ptr = start
while (Ptr != NULL)
Print Ptr → data.
Case 3 : If (start == NULL), print no element to display
else ptr = start
while (ptr → right != NULL)
Print ptr → data.

Date .....

Aim

Date ..... Expt. No ....9.... Page No ....18....

Aim : Create a doubly linked list of integers and display in forward & backward direction.

Program :

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
struct node
{
 struct node *left ;
 int data ;
struct node *right ;
 } ;
struct node * start = NULL , *ptr , *temp ;
void main ( ) ;
{
  int ch , item ;
 clrscr ( ) ;
while (1)
{
  printf ("\n 1. add \n 2. Display forward \n 3. Display
  backward \n 4. exit \n" ) ;
printf ("enter ur choice" ) ;
scanf ("%d", &ch ) ;
 switch (ch)
 {
```

Teacher's Signature: .....

case 4 : Exit.

Step 8 : Stop.

Output :

1. Add
2. Display forward
3. Display Backward
4. Exit.

Enter ur choice : 3

Enter the item : 45, 38

38 ⟶ 45 ⟶ NULL

```
case 1 :
printf ("enter the item");
scanf ("%d", &item);
temp = (struct node*) malloc(sizeof(struct node));
temp → data = item ;
temp → left = NULL ;
temp → right = NULL ;
If (start == NULL)
start = temp ;
else
{
 Start → left = temp ;
temp → right = start ;
start = temp ;
}
break ;
case 2 :
If (start == NULL)
printf ("no element to splay");
else
{ ptr = start ;
while (ptr != NULL)
{
printf ("%d →", ptr → data);
 ptr = ptr → right ;
```

```
}
  }
    break;
  case 3:
  if (start == NULL)
  printf("no element to display");
  else
  {   ptr = start;
    while (ptr -> right != NULL)
      ptr = ptr -> right;
    while (ptr != NULL)
  {
      printf("%d ->", ptr -> data);
        ptr = ptr -> left;
    }
  }
    break;
  case 4:
    exit (0);
  }
}
```

Teacher's Signature: .....

Algorithm :

step 1 : start
step 2 : define structure struct with members coef and
exp of type int polynomial
step 3 : Declare polynomial array P_1, P_2, and P_3
step 4 : Read m
step 5 : Call function read (P_1, m)
step 6 : read n
step 7 : call function read (P_2, n)
step 8 : set i = j = k = 0
step 9 : Repeat step 10 to 13 while i<m && j<n
step 10 : if (P_1[i].exp > P_2[j].exp)
set P_3[k++] = P_1[i++]
step 11 : else if P_1[i].coef + P_2[j].coef != 0
set P_3[k] = P_1[j++]
step 12 : else if (P_1[i].coef + P_2[j].coef != 0
Let P_3[k++].coef += P_2[j++].coef

step 13 : else
set i = i+1 and j = j+1
step 14 : while i<m then set P_3[k++] = P_1[i++]
step 15 : while j<n then set P_3[k++] - P_2[j++]
step 16 : Call function print (P_1, m),
Print (P_2, n) and print (P_3, k)
Step 17 : Stop.

Date ..... Expt. No .....10..... Page No .....21.....

Aim : Addition of 2 polynomials using array

Program :

```
#include <stdio.h>
#include <conio.h>
typedef struct
{
int coef, exp ;
}
  polynomial ;
void read (polynomial *, int );
void print (polynomial *, int);
 void main ()
{
Polynomial P1[10], P2[10], P3[20] ;
int m, n, i, j, k ;
Printf("number of terms for first polynomial :");
Scanf ("%d", &m);
printf("Enter %d terms in first polynomial :\n", m);
read (P1, m);
printf ("Number of terms in second polynomial:\n");
Scanf ("%d", &n);
print f("Enter %d term after second polynomial:\n", n);
 read (P2, n);
i = j = k = 0
```

Teacher's Signature: .....

void read (polynomial P[], int n)

1. start
2. Repeat step3 to 5 for i = 0 to n
3. Read P[i] . coef
4. Read P[i] . exp
5. set i = i+1
6. stop.

print (polynomial P[], int n)

1. start
2. if P[0] . exp == 0 then
   print P[0] . coef
3. else if P[0] . exp == 1 then
   print P[0] . coef 'x'
4. else
   print P[0] . coef "x^" . P[0] . exp
5. repeat. step 6 to 10 for i = 1 to n
6. if P[i] coef >0 then print '+'
7. if P[i] exp == 0 then
   Print P[i] coef
8. else if P[i] exp == 1 then
   Print P[i] coef "x"
9. else
   Print P[i] coef "x" P[i] exp
10. set i = i+1
11. Stop

```
while (i<m && j<n)
  {
    if (P1[i].exp > P2[j].exp)
  {
    P3[k++] = P1[k++] :
  }
else if (P1[i].exp < P2[j].exp)
  {  P3[k++] = P2[j++];
  }
else if (P1[i].coef + P2[j].coef != 0)
  {
    P3[k] = P1[i++];
    P3[k++].coef += P2[j++].coef ;
  }
else
  {
    i++ , j++ ;
  }
}
while (i<m)
{
  P3[k++] = P1[i++];
}
}
while(j<n)
{
```

Output :

```
number of terms in first polynomial : 4
 Enter 4 terms for first polynomial :
  coefficient : 5
  exponent    : 4
  coefficient : 6
  exponent    : 2
  coefficient : 10
  exponent    : 1
  coefficient : -6
  exponent    : 0
number of terms in second polynomial : 4
 Enter 4 terms for second polynomial :
coefficient : 7
exponent    : 3
coefficient : 3
exponent    : 2
coefficient : 2
exponent    : 1
coefficient : 7
exponent    : 0
First polynomial  : 5x^4 + 6x^2 + 10x - 6
Second polynomial : 7x^3 + 3x^2 + 2 + 7
 polynomial       : 5x^4 + 7x^3 + 9x^2 + 12x + 1
```

```
P3 [k++] = P2 [j++] ;
  }
printf ("\n first polynomial :");
 printf ( P1 , m);
Printf ("\n second polynomial :");
 Printf (P2 , n);
 printf ("\n polynomial Sum :");
 printf (P3 , k);
 getch ();
 }
Void read ( Polynomial P[], int n)
{
 Int i ;
for (i=0; i<n ; i++)
{   printf ("\n coefficient :");
  Scanf ("%d", & P[i] , coef );
  printf ("~~%d~~ exponent" );
  Scanf ("%d ", & P[i] . exp);
 }
}
Void print ( Polynomial P[] , int n )
{ Int i ;
If ( P[0] . exp == 0)
{ printf ("%d", P[0] . coef); }
else if (P[0] exp == 1)
```

Teacher's Signature: .....

```
{ printf ("%dx", P[0].coef ); }
  for (i=0; i<n ; i++)
  {
  printf ("+");
  } }
  Printf (P[i].exp==0)
  {
  printf ("%d", P[i].coef );
  }
else if (P[i].exp==1)
  {
  printf ("%dx", P[i].coef );
  }
  else
  {
  printf ("%dx^%d", P[i] coef , P[i].exp);
  }
  }
}
```

Teacher's Signature: .....

Algorithm :

Step 1 : start
step 2 : globally declare array stack [100], choice, n, top
x, i
step 3 : set top = -1
step 4 : Read n for enter the size of stack
step 5 : Read choice for enter the choice
step 6 : If choice = 1 call push(), If choice = 2 call pop()
If choice = 3 call display(), If choice = 4 then exit
step 7 : start switch statement.
case 1 : push case 2 : pop
case 3 : Display case 4 : exist.
step 8 : If default print 'please entes valid choice'
step 9 : while choice != 4 then return 0
step 10 : If (top >= n-1) print stack is overflow
If (top <= n-1) print stack is under flow
If (top >= 0) print the elements in stack
else print the stack is empty.
Step 11 : Stop.

Date ..... Expt. No .....11..... Page No .....25.....

Aim : To Implement Stack using array

Program :

```
#include <stdio.h>
#include <conio.h>
int stack [10], choice, n, top, x, i;
void push (void);
void pop (void);
void display (void);
int main ()
{
 clrscr();
 top = -1 ;
printf ("\n enter size of stack :");
scanf ("%d", &n);
printf ("\n\t stack operations using array");
Printf ("\n\t ..... ");
Printf ("\n\t 1. PUSH \n\t 2. POP \n\t 3. Display \n\t
        4. exit");
do
{   printf ("enter the choice");
    scanf ("%d", & choice
    Switch (choice)
{   case 1;
  {   push ();
```

Teacher's Signature: .....

Output :

1. Push
2. Pop
3. Display
4. Exist

Enter the choice : 1
Enter a value to be pushed : 12, 56
Enter the choice : 2
Popped element is 56
Enter the choice : 3
Elements in stack 12

```
        break;
    }
    case 2:
    {
        Pop();
        break;
    }
    case 3:
    {
        Display();
        break;
    }
    case 4:
    {   printf("\n\t Exit point");
        break;
    }
    default;
    {   printf("\n\t please enter a valid choice(1|2|3|4)");
    }
}
}   while (choice != 4);
    return 0;
}
void push()
{   if (top >= n-1)
```

```
{ printf ("\n It stack is over flow");
  }
  else
{ printf ("Enter a value to be pushed :");
  scanf ("%d", &x);
    top++;
  Stack [top = x ;
  }
} void pop()
  { if (top<=-1)
  { printf("\n It stack is under flow");
  }
  else
{ printf ("\n It the popped element is %d", stack[top]);
    top--;
  }
}
 void display()
{ if (top>=0)
{   printf ("\n the elements in stack \n");
  for (i = top; i>=0 ; i--)
  Printf ("\n %d ", stack [i]);
  Printf ("\n press next choice");
} else
  { printf ("\n the stack is empty");
} }
```

Teacher's Signature: .....

Algorithm :

Step 1 : start

Step 2 : Include all the header files which are used in the program & declare defined functions.

Step 3 : Define a Node with two members data & Next

Step 4 : Define a Node pointer 'top' & set it to NULL

Step 5 : Declare variables, choice, value.

Step 6 : while(1), print the stack using linked list.

Step 7 : Print the MENU : 1. Push
2. Pop
3. Display
4. Exit.

Step 8 : Read 'choice' to enter your choice.

Step 9 : start the switch statement.

Case 1 : push

Case 2 : Pop

Case 3 : Display

Case 4 : Exit

Step 10 : default, print wrong selection.

Step 11 : stop.

Date ..... Expt. No .....12..... Page No .....28.....

Aim : To Implement stack using linked list.

Program :

```
#include <stdio.h>
#include <conio.h>
 struct node
{  int data;
 struct node *next;
 {
   *top = NULL;
void push (int);
void pop();
void display();
void main()
{
 int choice, value;
 clrscr();
printf ("\n :: stack using linked list ::\n");
while (1)
{ printf ("\n ******* MENU *******\n");
 Printf ("1. push\n2. pop\n 3. Display\n 4. Exit\n ");
 printf (" Enter your choice: ");
 scanf ("%d", & choice );
 Switch (choice)
case 1 : printf ("enter the value to be insert :");
```

Teacher's Signature: .....

Void push (int value)

step 1 : start

step 2 : create new node → data = value.

step 3 : If (top = = NULL), new Node→next = NULL

step 4 : else Newnode → next = top

top = newnode

step 5 : print insertion is success.

step 6 : stop

Void Pop ()

step 1 : start

step 2 : create struct node ~~*temp~~ ~~=top~~

step 3 : If (top = = NULL), print stack is empty

step 4 : else struct node *temp = top

Print deleted elements.

step 5 : top = temp → next

free (temp)

step 6 : stop

Void display()

Step 1 : Start

Step 2 : If (top = = NULL) print stack is empty

step 3 : else struct node * temp = top

step 4 : while (temp → next != NULL)

step 6 : print the elements.

step 7 : stop

```
scanf ("%d", &value);
push (value);
break;
case 2: pop();
        break;
case 3: display();
        break;
case 4: exit(0);
default: printf("In wrong selection!!! please try
        again !!!\n");
  }
}
}
Void push (int value)
{ struct node *newnode;
newnode = (struct node*) malloc (size of (struct node));
Newnode → data = value;
If (top == NULL)
Newnode → next = NULL;
else
Newnode → next = top;
  top = newnode;
Printf ("\n Insertion is success !!!\n");
}
Void pop()
```

Teacher's Signature: .....

Output :

****** MENU ******

1. Push
2. POP
3. Display
4. Exit

Enter your choice : 1

Enter the value to be Insert. : 25, 8

Item Pushed.

Enter your choice : 3

Enter the ~~choice~~

Display the Stack : 25 → 8

```
{   If (top == NULL)
    printf ("\n Stack is empty !!!\n");
    else
{   struct node *temp = top;
    printf ("\n deleted element : %d", temp -> data);
    top = temp -> next;
    free (temp)
    }
}   void display()
{   If (top == NULL)
        Print f ("\n Stack is empty !!!\n");
    else
{   Struct node *temp = top;
    while (temp -> next != NULL)
{   printf ("%d --->", temp -> data);
    temp = temp -> next;
    }
Printf ("%d ---> NULL", temp -> data);
    }
}
```

Teacher's Signature: .....

Algorithm :

Step 1 : start
step 2 : Declare the variable top = -1 and array stack [20] and character variables *e and an array exp [20]

Step 3 : Declare Integer variable n_1, n_2, n_3, num
Step 4 : Read the expression from the user into exp
step 5 : set e = exp
Step 6 : check (*e != '\0') true, then repeat the Step 7 - 14
Step 7 : If *e is digit then set num = *e - 48 and push (num)
otherwise set n_1 = pop(), n_2 = pop()
step 8 : start a switch statement with *e a the expression.
step 9 : If case '+' then, set $n_3 = n_1 + n_2$ & break
Step 10 : If case '-' then, set $n_3 = n_2 - n_1$ & break
Step 11 : If case '*' then, set $n_3 = n_1 * n_2$ & break
Step 12 : If case '/' then, set $n_3 = n_2 / n_1$ & break
step 13 : then push (n_3)
step 14 : set e++
Step 15 : Print the result of expression
Step 16 : Stop

Date ..... Expt. No .....13..... Page No .....31.....

Aim : Evaluation of postfix Expression.

Program :

```
#include <stdio.h>
#include <conio.h>
int stack [20];
 int top = -1;
void push (int x)
{  struct [++ top] = x;
  }
   int pop()
{  return Stack [top--];
  }
   int main()
{  char exp[20];
     ch as *e ;
int n1, n2, n3, num ;
 clrscr ();
printf (" enter the expression :: ");
Scanf (" %s ", exp);
  e = exp ;
while ( *e != '\0')
 {  if (is digit (*e))
 {  num = *e - 48
      push (num)
```

Teacher's Signature: .....

Output :

Enter the expression : 1 2 3 4 5 * + * +)

The result of expression :

1 2 3 4 5 * + * +) = 47

```
}
  else
{ n1 = pop();
  n2 = pop();
  switch(*e)
{ case '+':
{ n3 = n1 + n2;
    break;
}
case '-':
{ n3 = n2 - n1;
    break;
}
case '*':
{ n3 = n1 * n2;
    break;
}
} push(n3)
} e++;
} printf("In the result of expression %s
= %d \n\n", exp, pop());
return 0;
}
```

Algorithm :

step 1 : start

step 2 : Declare variables queue (n), ch = a, front = 0
rear = 0, i, j = 1, x = n

step 3 : print queue using array. & 1. Insert 2. deletion
3. display & Exit.

step 4 : Read ch to enter the choice.

step 5 : start switch Statement.

case 1 : if (rear = = x) print queue is full
else print enter the number

case 2 : if (front = = rear) then queue is empty
else print deleted elements

case 3 : if (front = = rear) queue is empty
else print the elements

case 4 : exit.

default : print wrong choice.

step 6 : Stop.

Date ..... Expt. No ....14.... Page No ....33....

Aim: Implement queue using array

Program:

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#define n 5
Int main()
{
  int queue[n], ch=1, front=0, rear=0, i, j=1, x=n;
  clrscr();
  printf("queue (Using array)
  printf ("\n 1 Insertion\n2. deletion\n 3. Display.
          4. Exit");
while (ch)
{  printf("\n enter the choice");
   scanf("%d", &ch);
   switch (ch)
{
case 1:
if (rear==x)
print f("\n queue is full");
else
{ printf ("\n enter no %d:", j++)
scanf ("%d", & queue [rear++]);
```

Teacher's Signature: .....

Output :

Queue using Array

1. Insertion
2. Deletion
3. Display
4. Exit.

Enter the choice : 1

Enter no 1 : 12

Enter the choice : 1

Enter no 2 : 67

Enter the choice : 2

Deleted element is 12

Enter the choice : 3

Queue Elements are : 67

```
}
  break;
case 2:
 If(front == rear)
{   printf("\n queue is empty");
  }
else
{   printf("\n deleted element is %d", queue
    [front++]);
    x++ ;
}
break;
Case 3:
printf("\n queue element are :\n");
  If (front == rear)
  printf("\n queue is empty");
  else
{  for(i= front ; i < rear ; i++)
  {  printf("%d", queue[i]);
    printf("\n");  }
  break;
Case 4:   exit(0);
  default: printf("wrong choice: please see option");
   }
  }
} return 0;
}
```

Teacher's Signature: .....

Algorithm :

Step 1 : start

Step 2 : Define a node structure with two elements data & next and declare defined function

Step 3 : Define a node pointer front & rear

Step 4 : Implement main method by displaying Menu with list of operations & make suitable functions calls.

Step 5 : Declare variables choice, value

Step 6 : print MENU with list 1. Insert
2. Delete
3. Display
4. Exit

step 7 : Read choice to enter the choice.

Step 8 : Start Switch Statement

Case 1 : Read value to insert value.
Insert the value.

Case 2 : Delete

Case 3 : Display

Case 4 : exit.

Default : print wrong selection

Step 9 : Stop.

Date ..... Expt. No .....15..... Page No .....35.....

Aim : Implement Queue using linked list.

Program :

```
#include <stdio.h>
#Include <conio.h>
Struct Node
{
  Int data
Struct Node *next ;
};  *front = NULL, *rear = NULL ;
void Insert (int);
void delete ();
void display ();
void main ()
{
  Int choice, value;
  clrscr();
Printf ("\n:: queue Implementation using linked list::\n");
while (1)
{ printf ("\n ***** MENU ***** \n");
  Printf ("1. Insert \n2. Delete \n 3. Display \n 4. Exit \n");
  Printf ("Enter your choice:");
  Scanf("%d", &choice);
  Switch (choice)
{   case 1: printf ("enter the value to be Insert :");
```

Teacher's Signature: .....

Void Insert ()
step 1 : Start
step 2 : Create a new node
step 3 : set newnode → data = value
newnode → next = NULL
step 4 : If (front == NULL)
front = rear = newNode.
step 5 : Insert the element.
step 6 : stop

Void delete ()
step 1 : start
step 2 : if (front == NULL) queue is empty
else *temp = front.
Step 3 : set front = front → next
step 4 : Delet the element.
Step 5 : stop

Void Display ()
Step 1 : start
Step 2 : If (front == NULL) Queue is empty
else *temp = front
Step 3 : set while temp → next != NULL
then print temp → data.
Step 4 : set temp = temp → next
step 5 : Display the elements
Step 6 : Stop.

```
scanf("%d", &value);
insert (value);
break;
case 2 : delete (); break ;
case 3 : Display() ; break ;
case 4 : exit (0) ;
default : printf ("\n wrong selection!!! please try again!!\n");
}
}
}
void Insert (int value)
{
Struct Node *newNode ;
newNode =(struct Node *)malloc (size of (struct Node));
new Node → data = value ;
new Node → next = NULL ;
if (front = = NULL)
front = rear = newNode ;
else
{
rear → next = NewNode ;
rear = newNode ;
}
Printf ("\n Insertion is success !!! \n");
}
```

Output :

Queue Implimentation Using linked list.

```
****** MENU ** ****
1. Insert
2. Delete
3. Display
4. Exit

Enter your choice : 1
    Enter value to be Insert : 25, 36

Enter your choice : 2
    Enter value to delete : 25
Deleted element 25
```

```
void delete()
{
  if (front == NULL)
  printf ("\n queue is empty !!!\n");
else {
  Struct Node *temp. = front ;
  front = front → next ;
  Printf ("\n Deleted element : %d \n", temp → data );
  free (temp) ;
   }
}
void display ()
{
  if (front == NULL)
  Printf ("\n Queue is empty !!!\n");
else {
  Struct Node *temp = front ;
while (temp → next != NULL) {
  printf ("%d -->", temp → data );
  temp = temp -> next ;
  }
printf ("%d --> NULL \n", temp → data );
  }
}
```

Teacher's Signature: .....

Algorithm :

Step 1 : start
Step 2 : Define all user defined functions
Step 3 : Declare an Integer variable Item.
Step 4 : set head = NULL & update head as
head = Create (head, 10), head = Create
(head, 20), head = Create (head, 30)
head = Create (head 40)
Step 5 : Traverse the tree in any order such as
preorder, postorder, inorder.
Step 6 : Read the search Item from the Integer
value variable Item.
Step 7 : set found = search (head, Item)
Step 8 : check if (found) true, then print value
found otherwise print value not found
Step 9 : Stop.

Date ..... Expt. No ..... 16 ..... Page No ..... 38 .....

Aim : Search an element in binary search Tree.

Program :

```
#Include <stdio.h>
#Include <stdlib.h>
Struct BST node
{
   Int data
  Struct BST Node *left ;
  Struct BST node * right ;
  }
 Struct BST node *new node (Int data)
{ Struct BST node *t ;
   t = (struct BST node *) malloc (size of (struct
        BST node ))
   t → data = data ;
   t → left = t → right = NULL ;
   Return t ;
  }
Int Search (Struct BST node *head , Int val)
{ while (head != NULL)
  {
    If (val > head → data)
       head = head → right ;
```

Output :

10 → 20 → 30 → 40

Enter the element to search : 30

30 value is found in the tree.

```
        else if (val < head->data)
                head = head->left;
        else return 1
    }
    struct BST node *Create (struct BST node
        *head, int data)
    {   if (head == NULL)
    {  head = new node (data),
        return head
    }
else if (data ≤ head->data)
    head->left = Create (head->left, data);
else {
    head->right = Create (head->right->data)
    return head;
}
void preorder (struct BST node *head)
{
 if (head == NULL) return;
 printf ("%d ->", head->data);
 preorder (head->left);
 preorder (head->right);
}
int main ()
{
```

```
int Item;
Struct BST node *head = NULL;
  head = (reate (head , 10)
  head = (reate (head, 20)
  head = (reate (head, 30)
  head = (reate (head , 40)
  preorder (head);
  puts ("\n");
printf ("enter the element to search:");
  scanf ("%d, & Item);
int found = search (head, Item)
  If (found)
printf ("%d.value is found in the tree", Item);
  else
  printf ("%d value not found", Item);
}
  return 0;
}
```

Teacher's Signature: .....

Algorithm :

step 1 : start

step 2 : Define an array with size 10 and integer variables i, j, temp, n.

step 3 : Read the numbes of elements in n

Step 4 : start a loop with i = 0 and repeat the step 5 until i<n

Step 5 : Read the element & store it in array[i] respectively

Step 6 : start a loop with i = 0 and repeat the step 7 to 8 until i<n-1

step 7 : start an inner loop with j = i+1 and repeat step 8 until j<n

step 8 : check whether array[i] > array[j] If it is true then swap the elements

Step 9 : Print sorted array with start a loop with i = 0 and print the element in array[i] until i<n

Step 10 : Stop.

Date ..... Expt. No .....17..... Page No .....41.....

Aim : Implement Exchange Sort

Program :

```
#include <stdio.h>
int main()
{
    int array[10], i, j, temp, n;
printf("Enter the number of elements");
scanf("%d", &n);
printf("Enter %d numbers : \n");
    for (i=0; i<n; i++)
scanf("%d", &array[i]);
    for (i=0; i<n-1; i++)
{
    for (j=i+1; j<n; j++)
{
    if (array[i] > array[j])
{
    temp = array[i];
    array[i] = array[j];
    array[j] = temp;
}
}
}
```

Teacher's Signature: .....

Output :

Enter the number of element : 5

Enter 5 numbers : 8
1
3
10
6

sorted array is : 1
3
6
8
10

```
printf ("Sorted array is : \n");
    for (i=0 ; i<n ; i++)
printf ("%d \n", array[i]);
return 0;
}
```

3.

Teacher's Signature: .....

Algorithm :

step 1 : Start

step 2 : Declare Integer variables i, j, temp, n
mes, length and initialize length = 5

step 3 : Declare Integer array arr with size 5

step 4 : Read the numbes of elements from the
user & scan it into n

step 5 : Read the element from the user and
it in the array through a for loop

step 6 : Start a loop with i = 0 & Repeat step 7
until i<n

step 7 : scan the number into respective arr[i]

step 8 : start a loop with i=0 & Repeat step 9-10
until i<n-1

step 9 : Initialize min = i and start a inner loop
with j = i + 1 & repeat the step 10 until j<n

step 10 : check whether arr[j] < arr[min]
If it true then Initialize min = j and
then swap the element arr[min] of arr[i]

step 11 : print the Sorted array with start a loop

Aim: Implement selection sort.

Program:

```
#include <stdio.h>
int main()
{
  int arr[5], length=5, i, j, temp, n, min;
 printf("enter the number of elements");
 scanf("%d", &n);
 printf("Enter %d numbers", n);
 for (i=0; i<n; i++)
 scanf("%d", &arr[i]);
for (i=0; i<n-1; i++)
{
  min = i
  for (j=i+1; j<n; j++)
  if (arr[j]< arr[min])
  min = j;
  temp= arr[min];
  arr[min] = arr[i];
   arr[i] = temp;
 }
  printf("sorted array is ");
  for (i=0; i<n; i++)
```

Teacher's Signature: ....

i=0 and print the element is arr[i] until i<n

step 12 : stop.

Output :

Enter number of element : 5

Enter 5 number : 3
8
1
6
10

Sorted array is : 1
3
6
8
10

```
    Printf("%d\n", arr[i]);
    return 0;
}
```

Algorithm:

Step 1 : start

Step 2 : Declare Integer variables i, j, n, key, length & initialize length = 5

Step 3 : Declare an Integers array arr with size 5

Step 4 : Read the number of elements from the user and scan into n

Step 5 : Read the number from the user and scan it into the array through a loop

Step 6 : start a for loop with i = 0 and Repeat the step 7 until i < n

Step 7 : Scan the element into arr[i] respectively

Step 8 : Start a loop with i = 1 & repeat the step 9-11 until i < n

Step 9 : Assign key = arr[i] and j = j - 1

Step 10 : check (key < array[j] && j >= 0)

If it is true then assign arr[j+1] = arr[j] and --j

Step 11 : Assign arr[j+1] = key

Step 12 : Print the Sorted array with start a loop with i = 0 & repeat step 13 until i < n

Date ..... Expt. No .....19..... Page No .....45.....

Aim : Implement Insertion Sort.

Program :

```
#Include <stdio.h>
#Include <conio.h>
Void Insertion Sort (Int a[], Int n)
{
   Int j, p;
   Int tmp;
  for (p= 1 ; p<n; p++)
  }
Int main ()
{
  Int arr[5], length =5, i, j, key
  printf ("enter the number of elements");
  Scanf (" %d", &n);
  printf (" %d" enter %d number \n", n)
      for (i=0; i<n ; i++)
{
  key= arr[i];
  j = i - 1
while (key < arr[j] && j >=0)
{
  arr[j+1] = arr[j];
```

Teacher's Signature: .....

Step 13 : print the element arr[i] respectively

Step 14 : stop

Output :

Enter the number of elements : 5

Enter 5 numbers : 4
10
1
7
45

Sorted array is : 1
4
7
10
45

```
        --j;
    }
    arr[j+1] = key;
}
printf("Sorted array is \n");
for(i=0;i<n;i++)
printf("%d", arr[i]);
return 0;
}
```

Algorithm :

step 1 : Start
step 2 : Declare Integer variables i, count and an
array number with size 25
step 3 : Read the number of elements from the
user and scan it into count
step 4 : Read the number from the user
through an for loop
step 5 : Start a loop with i = 0 and repeat the
step 6 until i < count
step 6 : Scan the number into number[i] respectuly
step 7 : set first = 0 & last = count - 1
step 8 : check first < last, if is true then set
pivot = first, i = first, j = last
step 9 : check i < j, if it is true then check
number[i] <= number[pivot] && i < last
If true then i++ & check number[j]
> number[pivot] it is true then j--
step 10 : check i < j if it is true, then swap
the elements number[i] & number[j]
step 11 : set temp = number[pivot]
number[

Date ..... Expt. No ....20..... Page No ....47....

Aim : implement Quick sort

Program :

```
#include <stdio.h>
void quicksort (int number[25], int first, int last)
{
   int i, j, pivot, temp;
   if (first < last)
   {  pivot = first;
      i = fisst;
      j = last;
   {while (i<j)
   {
   while (number[i] > number[Pivot]
   j--;
   if (i<j)
   {  temp = number[i];
      number[i] = number[j];
      number[j] = temp;
   quicksort (number, first, j-1);
   quicksort (number, j+1, last);
   }}
int main ()
{  int i, count, number[25];
```

Teacher's Signature: .....

number[j] = temp

step 12 : set last = j-1 & repeat step 8-10

step 13 : set first = j+1 & repeat step 8-10

step 14 : print the sorted array through a loop
with i=0 & print the element number[i]
respectively until i< count

step 15 : stop

Output :

Enter the number of elements : 5

Enter 5 numbers : 4
10
3
9
5

Sorted order array is : 3 4 5 9 10

```
printf ("Enter the number of elements :");
scanf ("%d", &count);
printf ("Enter %d elements :", count);
} for (i=0 ; i<count ; i++)
scanf ("%d", &number[i]);
quicksort (number, 0, count-1);
printf ("sorted array is :");
for (i=0; i<count ; i++)
printf ("%d", number[i]);
return 0;
}
```