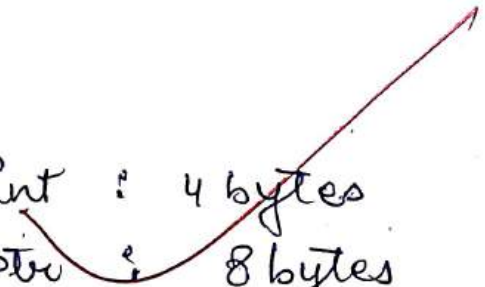
Output :

size of int : 4 bytes
size of ptr : 8 bytes
size of long : 8 bytes



Name ..... Class ..... Som ..... S.No. .....

Lab - 1

Aim - Write a C program to display size of int, point, long

```
#include <stdio.h>
int main()
{
    int a;
    int *ptr;
    long b;

    printf("Size of int : %lu bytes \n", sizeof(a));
    printf("Size of pointer : %lu bytes \n", sizeof(ptr));
    printf("Size of long : %lu bytes /n", sizeof(b));

    return 0;
}
```

10-11-25

Size of long : 8 bytes

INSTITUTE OF ENGINEERING AND TECHNOLOGY

(Mohanlal Sukhadia University, Udaipur)

Name ..... Class ..... Sem ..... S.No. .....

Experiment - 2

Aim :- Write a C program to print a 3X3 matrix using array.

```
#include <stdio.h>

int main()
{
    int matrix [3][3]
    int i,j ;

    printf ("Enter elements of 3x3 matrix:\n");
    for (i=0; i<3 ; i++) {
        for (j=0 ; j<3 ; j++){
            printf ("Enter element [%d][%d] :", i+1,j+1);
            scanf ("%d", &matrix[i][j]);

    }

    printf ("\n The 3x3 matrix is :\n");
    for (i=0; i<3 ; i++) {
        for (j=0 ; j<3; j++) {
            printf ("%d \t", matrix [i] [j]) ;
        }
        printf ("\n") ;
    }
    return 0;
}
```

Output :

```
Enter elements of 3X3 matrix :

Enter element [1][1] : 1
Enter element [1][2] : 2
Enter element [1][3] : 3
Enter element [2][1] : 4
Enter element [2][2] : 5
Enter element [2][3] : 6
Enter element [3][1] : 7
Enter element [3][2] : 8
Enter element [3][3] : 9

The 3X3 matrix is :
1 2 3
4 5 6
7 8 9
```

INSTITUTE OF ENGINEERING AND TECHNOLOGY

(Mohanlal Sukhadia University, Udaipur)

Name ..... Class ..... Sem ..... S.No. .....

Experiment -3

Aim :- write a program to print the transpose of 3x3 matrix using array.

```
#include <stdio.h>
int main () {
        int matrix [3][3]
        int transpose [3][3]
        int i,j;

        printf ("Enter elements of 3x3 matrix \n");
        for (i=0; i<3; i++) {
            for (j=0; j<3; j++) {
                printf ("Enter the element [%d][%d]: ",i+1,j+1);
                scanf ("%d", &matrix [i][j]);
            }
        }

        printf ("\n The 3x3 matrix is : \n");
        for (i=0; i<3; i++) {
            for (j=0; j<3; j++) {
                printf ("%d \t", matrix [i][j]);
            }
        printf ("\n");
        }
```

12-11-25

Enter elements of 3x3 matrix:

Enter element [1][1]: 1
Enter element [1][2]: 2
Enter element [1][3]: 3
Enter element [2][1]: 4
Enter element [2][2]: 5
Enter element [2][3]: 6
Enter element [3][1]: 7
Enter element [3][2]: 8
Enter element [3][3]: 9

The 3x3 matrix is:

1	2	3
4	5	6
7	8	9

The transpose matrix is:

1	4	7
2	5	8
3	6	9

```
for (i=0; i<3; i++) {
    for (j=0; j<3; j++) {
        transpose[j][i] = matrix[i][j];
    }
}

printf("\n The transpose of matrix is: \n");
    for (i=0; i<3; i++) {
        for (j=0; j<3; j++) {
            printf("%d \t", transpose[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

12-11-25

O/P :-

---- Stack Menu -----

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

Enter your choice : 1
Enter value to push : 10

10 pushed into stack

----- Stack menu - -----

1). Push
2). pop
3. peek
4. Display
5. Exit

Enter your choice : 1
Enter your choice : 20

20 pushed into Stack

---- Stack menu ---

1. Push
2. pop
3. Peek
4. Display
5. Exit

Enter your choice : 4
Stack Elements are :

Index 1 → value 20
Index 0 → Value 10

INSTITUTE OF ENGINEERING AND TECHNOLOGY

(Mohanlal Sukhadia University, Udaipur)

Name ..Lavina..Jamsoli..... Class ..... Sem ..... S.No. .....

Experiment - 4

Aim → Write a C Program to display all Stack Operations.

```
#include <stdio.h>
#include <conio.h>
#define SIZE 5

int stack [SIZE];
int top = -1;

void push (int value) {
    if (top == SIZE - 1)
        printf("Stack Overflow ! Cannot push %d\n", value);
    else {
        top ++;
        Stack [top] = value;
        printf ("%d pushed into stack.\n", value);
    }
}

void pop () {
    if (top == -1)
        print f ("Stack Underflow ! Stack is empty.\n");
    else {
        print f("%d popped from stack.\n", stack[top]);
```

----- Stack menu -----

1. Push
2. Pop
3. Peek
4. Display
5. Exit

Enter your choice : 2

So popped from stack

----- Stack menu ---

1. Push
2. pop
3. peek
4. Display
5. Exit

Enter your choice : 4

Stack element are

Index 0 → value 10

----- Stack menu ---

1. Push
2. pop
3. peek
4. Display
5. Exit

Enter your choice : 5

--- Existing Program ---

```
    }
}

void peek () {
    if (top == -1)
        printf ("Stack is empty ! No top element.\n");
    else
        printf("Top element is: %d\n", stack [top]);
}

void display () {
    int i;
    if (top == -1)
        printf ("Stack is empty.\n");
    else {
        printf ("Stack elements are :\n");
        for (i = top ; i >= 0 ; i--)
            printf ("Index %d -> Value %d\n", stack[i]);
    }
}

int main () {
    int choice, value ;
    clrscr ();

    Print f ("Simple Stack Program using Array\n");
    while (1)
    {
```

Name ..... Class ..... Sem ..... S.No. .....

```
printf("\n--- Stack Menu ---\n");
printf("\n1. Push\n2. Pop\n3. Peek\n4. Display\n5. Exit\n");
printf("Enter your choice : ");
Scanf("%d", &choice);

Switch (choice) {
  Case 1:
    printf("Enter value to push : ");
    Scanf("%d", &value);
    push (value);
    break

  Case 2:
    pop ();
    break;

  Case 3:
    peek ();
    break;

  Case 4:
    display ();
    break;

  Case 5:
    return 0;

  default :
```

INSTITUTE OF ENGINEERING AND TECHNOLOGY

(Mohanlal Sukhadia University, Udaipur)

Name ..... Class ..... Sem .....S.No. .....

```
        printf("Invalid choice! Please try again.\n");
    }
}
}
```

26/11/2025

Output :

---- Queue Menu ----

1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Exit

Enter your choice : 1

Enter value to Insert : 3

3 inserted into queue.

Name ..... Class ..... Sem ..... S.No. .....

Experiment - 5

Aim :- Write a program to display the operation of queue

```
# include < stdio. h>
# define Max 100

int queue [Max];
int front = -1 ;
int rear = -1 ;

void enqueue (int x) {
        if (rear == Max-1)
        {
          Printf ("Queue is full \n");
        }
        else {
          if (front == -1)
            front = 0 ;
            rear ++ ;
            queue [rear] = x ;
            queue Printf ("%d inserted into queue \n");
        }
    }
void dequeue () {
        if (front == -1 || front > rear)
```

```
{
    printf("Queue is Empty \n");
}
else {
    Printf("%d removed from queue \n", queue[front]);
    front++;
}
}
void peek() {
    if (front == -1 || front > rear) {
        printf("Queue is Empty \n");
}
else {
    printf("Front element : %d \n", queue[front]);
}
}
void display() {
    if (front == -1 || front > rear) {
        printf("queue is Empty \n");
}
else {
    printf("queue elements : ");
    for (int i = front; i <= rear; i++) {
        printf("%d", queue[i]);
}
    printf("\n");
}
}
```

```
int main() {
    int choice, value;

while(1) {
Printf("\n --- Queue Menu ---\n");
Printf("1. Enqueue \n");
Printf("2. Dequeue \n");
Printf("3. Peek \n");
Printf("4. Display \n");
Printf("5. Exit \n");
Printf("Enter your choice:");
Scanf("%d", &choice);

switch(choice) {
Case 1:
Printf("Enter value to Insert:");
Scanf("%d", &value);
Enqueue(value);
break;
Case 2:
dequeue();
break;
Case 3:
peek();
break;
Case 4:
display();
break;
```

Name ..... Class ..... Sem ..... S.No. .....

```
        case 5:
            return 0;
        default :
            printf ("Invalid choice \n");
    }
}
```

20/11/2025

Output :

1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Exit

Enter your choice : 1
Enter ~~your~~ value to insert : 12
12 inserted

INSTITUTE OF ENGINEERING AND TECHNOLOGY

(Mohanlal Sukhadia University, Udaipur)

Name ..... Class ..... Sem ..... S.No. .....

Experiment - 6

Write a program to display basic operation of circular queue

```
# include <stdio.h>
# define size 5

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

void enqueue (int value) {
    if ((front == 0 && rear == size-1) ||
                (rear+1 == front)) {
        printf ("queue is full! \n");
    }
    else {
        if (front == -1)
            front = 0;
        rear = (rear+1) % size;
        queue[rear] = value;
        printf ("%d inserted \n", value);
    }
}
void dequeue() {
    if (front == -1) {
        printf ("queue is empty! \n");
    }
```

```
else {
printf("%d deleted \n", queue[front]);
    if (front == rear) {
        front = rear = -1;
    }
else {
    front = (front + 1) % size;
    }
}
}
void peek() {
    if (front == -1) {
        printf("queue is Empty ! \n");
    }
else {
printf("front element : %d \n", queue[front]);
}
}
void display() {
    if (front == -1) {
        printf("queue is Empty ! \n");
    }
else {
printf("queue : ");
int i = front;
while (1)
{
```

```
        Printf ("%d", queue[i]);
        if (i == rear)
            break;
        i = (i+1) % size;
    }
    printf ("\n");
  }
}
int main() {
    int choice, value;
    while (1) {
        printf ("\n 1. Empty \n 2. Dequeue \n 3. Peek \n 4.
                Display \n 5. Exit \n");
        Printf ("Enter your choice: ");
        scanf ("%d", &choice);

switch (choice) {
    case 1:
        Printf ("Enter value to insert: ");
        Scanf ("%d", &value);
        Enqueue (value);
        break;

    case 2:
        Dequeue ();
        break;
```

```
case 3:
    peek();
    break;
case 4:
    display();
    break;
case 5:
    return 0;
default:
    Printf("Invalid choice!\n")
}
}
}
```

26/11/2025

Name Lavina Jamboli Class ..... Sem ..... S.No. .....

Experiment - 7

Write a program to display basic operation of singly linked list.

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

struct Node {
    int data;
    struct Node * Next;
};
struct Node * head = Null;
void Insert at Beginning (int value) {
  struct Node * newNode = (struct Node *) malloc
    (size of (struct Node));

  newNode → data = value;
  newNode → next = head;
    head = newNode;
    printf ("\n Inserted %d at beginning \n", value);
}
void Insert At End (int value) {
  struct Node * newNode = (struct Node *) malloc (size
  of (struct Node));
    New Node → data = value;
    New Node → next = NULL;
```

---- Singly linked list menu ----

1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Delete Node
5. Display list
6. Exit

Enter your choice : 1

Enter your value : 10

Inserted 10 at beginning.

---- Singly linked list menu ----

1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Delete Node
5. Display List
6. Exit.

Enter your choice : 1

Enter value : 20

Inserted 20 at beginning.

```
if (head == NULL) {
    head = newNode;
    printf ("Inserted %d at end \n", value);
    return;
}
struct Node * temp = head;
while (temp -> next != NULL) {
    temp = temp -> next;
}
temp -> next = newNode;
printf ("Inserted %d at the end \n", value);
}
void Insert_At_position (int value, int position) {
    struct Node* newNode = (struct Node*) malloc (sizeof (struct Node));
    New Node -> data = value;

if (position == 1) {
    newNode -> next = head;
    head = newNode;
    printf ("Inserted %d at position %d \n", value, position
    return;
}
struct Node * temp = head;
for (int i = 1; i < position - 1 && temp != NULL; i++) {
        temp = temp -> next;
    }
if (temp == NULL) {
printf ("Position out of range \n");
```

```
    free (newNode);
    printf ("Invalid
    return;
}
newNode -> next = temp -> next;
temp -> next = newNode;
printf ("Inserted %d at position %d \n", value, position);
}

// function to delete a node
void deleteNode (int value) {
    if (head == NULL) {
        printf ("List is empty \n");
        return;
    }

struct Node * temp = head;
// If head needs to be deleted
if (head -> data == value) {
    head = head -> next;
    free (temp);
    printf ("Deleted %d \n", value);
    return;
}

struct Node * prev = NULL;
while (temp != NULL && temp -> data != value) {
    prev = temp;
```

```
    temp = temp -> next;
}
if (temp == NULL) {
    printf("value %d not found \n", value);
    return;
}

prev -> next = temp -> next;
free(temp);
printf("Deleted %d \n", value);
}

// function to display list
void display() {
    if (head == NULL) {
        printf("List is empty \n");
        return;
    }

    struct Node *temp = head;
    printf("Linked list: ");
    while (temp != NULL) {
        printf("%d -> ", temp -> data);
        temp = temp -> next;
    }
    printf("NULL \n");
}
```

```
int main () {
int choice, value, position;

while (1) {
printf ("\n--- Singly linked list menu ---\n");
printf ("1. Insert at Beginning\n");
printf ("2. Insert at End\n");
printf ("3. Insert at Position\n");
printf ("4. Delete Node\n");
printf ("5. Display list\n");
printf ("6. Exit\n");
printf ("Enter your choice:");
scanf ("%d", &choice);

switch (choice) {
case 1:
    printf ("Enter value:");
    scanf ("%d", &value);
    InsertAtBeginning (value);
    break;

case 2:
    printf ("Enter value:");
    scanf ("%d", &value);
    printf ("Enter position:");
    scanf ("%d", &position);
    InsertAtPosition (value, position);
    break;
```

```
case 3:
    printf ("Enter value: ");
    scanf ("%d", &value);
    printf ("Enter position: ");
    scanf ("%d", &position);
    insertAtPosition(value, position);
    break;

case 4:
    printf ("Enter value to delete: ");
    scanf ("%d", &value);
    deleteNode(value);
    break;

case 5:
    display();
    break;

case 6:
    printf ("Exiting...\n");
    exit(0)

default:
    printf ("Invalid choice! Try again.\n");
}
}
return 0;
}
```

INSTITUTE OF ENGINEERING AND TECHNOLOGY
(Mohanlal Sukhadia University, Udaipur)

Name Tavina Jamboli Class ..... Sem ..... S. No .....

Experiment - 8

Write a program to Explain linear search operation.

```
#include <stdio.h>

int main()
{
int a[50], n, i, key, found = 0;
printf("Enter number of elements : ");
scanf("%d", &n);
printf("Enter the elements : \n");
for(i=0; i<n; i++)
{
scanf("%d", &a[i]);
}
printf("Enter element to search:");
scanf("%d", &key);
for(i=0; i<n; i++)
{
if(a[i] == key)
{
printf("Element found at position %d", i+1);
found = 1;
break;
}
}
```

Output :

Enter number of elements : 5
Enter the elements : 1 5 8 11 4
Enter the element to search : 8
Element found at position 3.

```
if (found == 0)
{
    printf ("Element not found");
}
return 0;
}
```

INSTITUTE OF ENGINEERING AND TECHNOLOGY
(Mohanlal Sukhadia University, Udaipur)

Name..... Class..... Sem..... S. No.....

Experiment - 9

Aim :- Write a program to display Binary search operation in an array

```
#include <stdio.h>

int main()
{
    int a[50], n, i, key, low, high, mid, found=0;
    printf("Enter number of elements : ");
    scanf("%d", &n);
    printf("Enter elements in sorted order : \n");
    for( i=0; i<n; i++)
    {
        scanf("%d", &a[i]);
    }
    printf("Enter element to search:");
    scanf("%d", &key);
    low = 0;
    high = n-1;

    while (low <= high)
    {
        mid = (low + high) / 2;
        if (a[mid] == key)
        {
            printf("Element found at position %d", mid+1);
```

Output :-

Enter number of elements : 6
Enter elements in sorted order : 1 3 5 7 9 11
Enter element to search : 9
Element found at position : 5

Name.....Class.....Sem.....S. No.....

```
        found = 1;
        break;
    }
    else if (a[mid] < key)
    {
        low = mid + 1;
    }
    else
    {
        high = ~~mid-1~~ mid - 1;
    }
}
    if (found == 0)
    {
        printf ("\n Element not found?!");
    }
    return 0;
}
```

INSTITUTE OF ENGINEERING AND TECHNOLOGY
(Mohanlal Sukhadia University, Udaipur)

Name....Lavina Jawtoli.....Class.....Sem.....S. No.....

Experiment - 10

Aim :- WAP to reverse a list using stack.

```
#include <stdio.h>
#define MAX 100          // max. size of stack
int stack[MAX];
int top = -1;
// function to push element onto stack
void push (int value)
{
   if (top == max-1)
      printf ("stack overflow\n");
   else
      stack[++top] = value;
}

// func^n to pop element from stack
int pop()
{
   if (top == -1) {
      printf ("stack Underflow\n");
      return -1;
   }
   else
      return stack[top--];
}
```

Output:

Enter number of element : 5
Enter elements : 1 4 7 8 9

Reversed list : 9 8 7 4 1

INSTITUTE OF ENGINEERING AND TECHNOLOGY
(Mohanlal Sukhadia University, Udaipur)

Name.....Class.....Sem.....S. No.....

```
int main()
{
  int list[MAX], n, i;
  printf("Enter number of elements: ");
  scanf("%d", &n);
  printf("Enter elements:\n");
  for(i=0; i<n; i++)
      scanf("%d", &list[i]);

// push all elements into stack
  for(i=0; i<n; i++)
     push(list[i]);

// pop all elements back into list (reversed)
  for(i=0; i<n; i++)
     list[i] = pop();
  printf("\n Reversed list: ");
  for(i=0; i<n; i++)
     printf("%d", list[i]);
  return 0;
}
```

INSTITUTE OF ENGINEERING AND TECHNOLOGY
(Mohanlal Sukhadia University, Udaipur)

Name Lavina Jamboli Class..... Sem..... S. No.....

Experiment - 11

Aim :- WAP to calculate factorial of a number using stack

```
# include <stdio.h>
# define MAX 50
int stack [MAX];
int top = -1;

void push (int x) {
    if (top == MAX - 1)
        printf ("Stack Overflow \n");
    else
    stack [++top] = x;
}
int pop() {
    if (top == -1)
        return -1;
    else
    return stack [top--];
}
int main() {
    int n, i;
    long long fact = 1;
    printf ("Enter a number: ");
    scanf ("%d", &n);

// push numbers 1 to n
```

Output:

Enter a number : 4
Factorial of 4 : 24

```
for (i=1; i<=n; i++) {
    push(i);
}
// pop and multiply
while (top != -1) {
    fact = fact * pop();
}
printf("\n factorial of %d = %ld\n", n, fact);

return 0;
}
```

21/01/26

Output :

Enter number of elements : 5
Enter the elements : 1 5 8 11 4
Enter the element to search : 8
Element found at position 3.

Name Lavina Jambali Class Sem S. No

Experiment - 8

Write a program to implement linear search operation

```
#include <stdio.h>

int main()
{
int a[50], n, i, key, found = 0;
printf("Enter number of elements : ");
scanf("%d", &n);
printf("Enter the elements : \n");
for (i=0; i<n; i++)
{
scanf("%d", &a[i]);
}
printf("Enter element to search:");
scanf("%d", &key);
for (i=0; i<n; i++)
{
if (a[i] == key)
{
printf("Element found at position %d", i+
found = 1;
break;
}
}
```

```
if (found == 0).
{
    printf ("\n Element not found");
}
return 0;
}
```

Output :-

Enter number of elements : 6
Enter elements in sorted order : 1 3 5 7 9 11
Enter element to search : 9
Element found at position : 5

INSTITUTE OF ENGINEERING AND TECHNOLOGY
(Mohanlal Sukhadia University, Udaipur)

Name.....Class.....Sem.....S. No.....

Experiment - 9

Aim :- Write a program to display Binary search operation in an array.

```
#include <stdio.h>

int main()
{
    int a[50], n, i, key, low, high, mid, found=0;
    printf("Enter number of elements : ");
    scanf("%d", &n);
    printf("Enter elements in sorted order : \n");
    for( i=0; i<n; i++)
    {
        scanf("%d", &a[i]);
    }
    printf("Enter element to search:");
    scanf("%d", &key);
    low = 0;
    high = n-1;

    while (low <= high)
    {
        mid = (low + high) / 2;
        if (a[mid] == key)
        {
            printf("Element found at position %d", mid+1);
```

```
found = 1;
break;
}
else if (a[mid] < key)
{
    low = mid + 1;
}
else
{
    high = ~~mid-1~~ mid - 1;
}
}
if (found == 0)
{
    printf ("\n Element not found?");
}
return 0;
}
```

Output:

Enter number of element : 5
Enter elements : 1 4 7 8 9

Reversed list: 9 8 7 4 1

Name Lavina Jandoli Class Sem S. No

Experiment - 10

Aim :- WAP to reverse a list using stacks.

```
#include <stdio.h>
#define MAX 100          // max. size of stack
int stack[MAX];
int top = -1;
// function to push element onto stack
void push (int value)
{
   if (top == max - 1)
      printf ("stack overflow \n");
   else
      stack[++top] = value;
}

// func^n to pop element from stack
int pop()
{
   if (top == -1) {
      printf ("stack Underflow \n");
      return -1;
   }
else
   return stack[top--];
}
```

```
int main()
{
    int list[MAX], n, i;
    printf("Enter number of elements: ");
    scanf("%d", &n);
    printf("Enter elements:\n");
    for(i=0; i<n; i++)
        scanf("%d", &list[i]);

// push all elements into stack
for(i=0; i<n; i++)
    push(list[i]);

// pop all elements back into list (reversed)
for(i=0; i<n; i++)
    list[i] = pop();
printf("\n Reversed list:");
for(i=0; i<n; i++)
    printf("%d", list[i]);
return 0;
}
```

Output:

```
Enter a number: 4
Factorial of 4 : 24
```

INSTITUTE OF ENGINEERING AND TECHNOLOGY

(Mohanlal Sukhadia University, Udaipur)

Name..... Lavina Jamboli .....Class.....Sem.....S. No.....

Experiment - 11

• Aim :- WAP to calculate factorial of a number using stack

⇒

```
# include <stdio.h>
# define MAX 50
int stack [MAX];
int top = -1;

void push (int x) {
    if (top == MAX - 1)
        printf ("Stack Overflow \n");
    else
    stack [++top] = x;
}
int pop () {
    if (top == -1)
        return -1;
    else
    return stack [top--];
}
int main () {
    int n, i;
    long long fact = 1;
    printf ("Enter a number: ");
    scanf ("%d", &n);

    // push numbers 1 to n
```

```
for (i=1; i<=n; i++) {
    push(i);
}

// pop and multiply
while (top != -1) {
    fact = fact * pop();
}
printf("Factorial of %d = %d\n", n, fact);

return 0;
}
```

21/01/26

Output :

Enter number of terms in the polynomial : 3
Enter coefficients power of x and power of y :

5 2 1
3 2 1
2 1 0

Polynomial Representation :

Coefficient	x - power	y - power
5	2	1
3	2	1
2	1	0

INSTITUTE OF ENGINEERING AND TECHNOLOGY

(Mohanlal Sukhadia University, Udaipur)

Name ..... Class ..... Sem ..... S.No. .....

Experiment - 12

- Aim :- Write a program to represent a 2-variable polynomial using Array.

```
#include <stdio.h>

int main() {
    int poly[20][3];
    int n, i;
    printf("Enter number of terms in the polynomial:");
    scanf("%d", &n);
    printf("Enter coefficient, power of x & power of y :\n");
    for (i=0; i<n; i++) {
        scanf("%d %d %d", &poly[i][0], &poly[i][1],
              &poly[i][2]);
    }
    printf("\n Polynomial representation :\n");
    printf(" coefficient  x- power  Y- power \n");
    for (i=0; i<n; i++) {
        printf("%d \t \t %d \t %d\n", poly[i][0],
               poly[i][1], poly[i][2]);
    }
    return 0;
}
```

INSTITUTE OF ENGINEERING AND TECHNOLOGY

(Mohanlal Sukhadia University, Udaipur)

Name ..... Class ..... Sem ..... S.No. .....

Experiment - 13

Aim:- WAP to represent Sparse Matrix using Array

```
#include <stdio.h>
int main() {
    int r, c, i, j;
    int mat[10][10];
    int sparse[50][3];
    int K = 1;
    printf("Enter the no. of rows & columns: ");
    scanf("%d %d", &r, &c);
    printf("Enter elements of the matrix :\n");
    for (i=0; i<r; i++) {
        for (j=0; j<c; j++) {
            scanf("%d", &mat[i][j]);
        }
    }
// first row stores rows, columns & non-zero count.
sparse[0][0] = r;
sparse[0][1] = c;
sparse[0][2] = 0;

// convert to sparse form
for (i=0; i<r; i++) {
    for (j=0; j<c; j++) {
        if (mat[i][j] != 0) {
            sparse[K][0] = i;
```

Output

Enter number of rows & columns : 3 3

Enter elements of the matrix : 0 0 1 2 0 0 0 0 3

Sparse Matrix Representation :

Row	Col	Value
3	3	3
0	2	1
1	0	2
2	2	3

```
        sparse[k][1] = j;
        sparse[k][2] = mat[i][j];
        k++;
        sparse[0][2]++;
      }
    }
  }
  printf("\n Sparse matrix Representation: \n");
  printf("Row col value \n");
  for(i=0; i<= sparse[0][2]; i++){
      printf("%d %d %d \n", sparse[i][0],
             sparse[i][1], sparse[i][2]);
  }
  return 0;
}
```

Output :-

-Enter number of elements : 5

-Enter the elements : 7 3 1 4 8

Sorted array : 1 3 4 7 8

INSTITUTE OF ENGINEERING AND TECHNOLOGY
(Mohanlal Sukhadia University, Udaipur)

Name : ..... Class ..... Sem ..... S. No. .....

Experiment - 14

Aim :- WAP to show the Bubble sort operation

```
#include <stdio.h>
int main()
{
int a[50], n, i, j, temp;
printf("Enter no. of elements: ");
scanf("%d", &n);
printf("Enter the elements: \n");
for(i=0; i<n; i++){
    scanf("%d", &a[i]);
}

// Bubble sort logic
for(i=0; i<n-1; i++){
    for(j=0; j<n-i-1; j++){
        if(a[j] > a[j+1]){
            temp = a[j];
            a[j] = a[j+1];
            a[j+1] = temp;
        }
    }
}
printf("sorted array: \n");
for(i=0; i<n; i++){
    printf("%d", a[i]);
}
return 0;
}
```

Output :

Enter number of elements : 5

Enter the elements : 4 6 1 9 7

Sorted Array : 1 4 6 7 9

INSTITUTE OF ENGINEERING AND TECHNOLOGY
(Mohanlal Sukhadia University, Udaipur)

Name : ..... Class ..... Sem ..... S. No. .....

Experiment - 15

Aim :- WAP to show the Insertion sort Operation.

```
# include <stdio.h>
int main() {
    int a[50], i, n, j, key;
    printf("Enter number of elements : ");
    scanf("%d", &n);
    printf("Enter the elements : \n");
    for (i=0; i<n; i++) {
        scanf("%d", &a[i]);
    }

// Insertion sort logic
for (i=1; i<n; i++) {
    key = a[i];
    j = i-1;
    while (j>=0 && a[j] > key) {
        a[j+1] = a[j];
        j--;
    }
    a[j+1] = key;
}
printf("sorted array : \n");
for (i=0; i<n; i++) {
    printf("%d", a[i]);
}
return 0;
}
```

INSTITUTE OF ENGINEERING AND TECHNOLOGY
(Mohanlal Sukhadia University, Udaipur)

Name..... Class..... Sem..... S. No.....

Experiment - 16

Aim:- Write a program to implement stack using linked list.

```
#include <stdio.h>
#include <stdlib.h>
// define node structure
struct Node {
int data;
struct Node *next;
};
struct Node *top = NULL;

// push operation
void push(int item) {
    struct Node *newNode;
    newNode = (struct Node *) malloc (sizeof(struct Node));

    if (newNode == NULL) {
        printf("Stack overflow\n");
        return;
    }
newNode -> data = item;
newNode -> next = top;
    top = newNode;
    printf("%d pushed into stack\n", item);
}

// pop operation
```

Output:

--- stack using linked list ---

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

Enter your choice : 1

Enter element to push : 5

5 pushed into stack

---- Stack using linked list

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

Enter your choice : 3

```
void pop() {
    struct Node * temp;
    if (top == NULL) {
        printf("Stack underflow \n");
        return;
    }
    temp = top;
    printf("popped element : %d\n", top->data);
    top = top->next;
    free(temp);
}
// Display Stack
void display()
    struct Node * temp;
    if (top == NULL) {
        printf("Stack is empty \n");
        return;
    }
    temp = top;
    printf("Stack elements : \n");
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL \n");
}
// Main function
int main() {
```

```
int choice, item;
while (1) {
    printf ("\n --- Stack using linked list ---\n");
    printf ("1. Push \n");
    printf ("2. pop\n");
    printf ("3. Display\n");
    printf ("4. Exit \n");
    printf ("Enter your choice");
    scanf ("%d", &choice);
switch (choice) {
case 1:
    printf("Enter element to push:");
    scanf ("%d", &item);
    push (item);
    break;
case 2:
    pop();
    break;
case 3:
    display();
    break;
case 4:
    exit(0)
default :
    printf ("Invalid choice \n");
}
}
return 0;
```

INSTITUTE OF ENGINEERING AND TECHNOLOGY
(Mohanlal Sukhadia University, Udaipur)

Name.....Class.....Sem.....S. No.....

Experiment - 17

Aim :- WAP to implement Queue using Linked list.

⇒
```
#include <stdio.h>
#include <stdlib.h>
// define node structure
struct node {
    int data;
    struct node * next;
};
struct node * front = NULL;
struct node * rear = NULL;

// Enqueue operation
void enqueue(int item) {
    struct Node * newNode;
    newNode = (struct Node *) malloc (sizeof(struct Node));
    if (newNode == NULL) {
        printf("Queue overflow\n");
        return;
    }
newNode -> data = item;
newNode -> next = NULL;

if (rear == NULL) {
    front = rear = newNode;
}
```

Output:

--- Queue using Linked List ---

1. Enqueue
2. Dequeue
3. Display
4. Exit

Enter your choice : 3

Queue elements :

2 → 6 → NULL

```
else {
    rear -> next = newNode;
    rear = newNode;
}
printf("%d Inserted into queue\n", item);
}

// Dequeue operation
void dequeue() {
    struct Node *temp;
    if (front == NULL) {
        printf("Queue Underflow\n");
        return;
    }
    temp = front;
    printf("Deleted element: %d\n", front->data);
    front = front->next;
    if (front == NULL) {
        rear = NULL;
    }
    free(temp);
}

// Display queue
void display() {
    struct Node *temp;
    if (front == NULL) {
        printf("Queue is empty\n");
        return;
    }
```

```
    temp = front;
    printf("Queue elements: \n");
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}
int main() {
    int choice, item;
    while (1) {
        printf("\n -- Queue using linked list --\n");
        printf("1. Enqueue\n");
        printf("2. Dequeue\n");
        printf("3. Display\n");
        printf("4. Exit\n");
        printf("Enter your choice");
        scanf("%d", &choice);
    switch (choice) {
    Case 1:
        printf("Enter element to insert: ");
        scanf("%d", &item);
        enqueue(item);
        break;
    Case 2:
        dequeue();
        break;
    Case 3:
        ~~display()~~
        break;
```

Name.....Class.....Sem.....S. No.....

```
display ();
break;
case 4:
    exit (0);
default :
    printf ("Invalid choice \n");
}
}
return 0;
}
```

Experiment - 18

Aim:- WAP for BFS using Adjacency Matrix

```
#include <stdio.h>
int n;
int adj[20][20];
int visited[20];
int queue[20], front = -1, rear = -1;

void BFS (int start)
{
  int i, v;
  printf("%d", start);
  visited[start] = 1;
  queue[++rear] = start;
  while(front != rear)
  {
    v = queue[++front];
    for(i=0; i<n; i++)
    {
      if(adj[v][i] == 1 && visited[i]==0)
      {
        visited printf("%d", i);
        visited[i] = 1;
        queue[++rear] = i;
      }
    }
  }
}
```

Output :

Enter number of vertices : 3

Enter adjacency matrix : 1 2 3
4 5 6
7 8 9

Enter starting vertex (0 to 2) : 0

BFS traversal : 0

```
int main() {
int i, j, start;
printf("Enter number of vertices: ");
scanf("%d", &n);
printf("Enter adjacency matrix:\n");
for(i=0; i<n; i++)
    for(j=0; j<n; j++)
        scanf("%d", &adj[i][j]);
for(i=0; i<n; i++)
    visited[i] = 0;
printf("Enter starting vertex (0 to %d): ", n-1);
scanf("%d", &start);
printf("BFS Traversal: ");
BFS(start);
return 0;
}
```

INSTITUTE OF ENGINEERING AND TECHNOLOGY

(Mohanlal Sukhadia University, Udaipur)

Name.....Class.....Sem.....S. No.....

Experiment - 19

Aim :- WAP to Implement DFS using Adjacency Matrix

⇒

```
#include <stdio.h>
int n;
int adj[20][20];
int visited[20];

void DFS(int v)
{
   int i;
   printf("%d ", v);
   visited[v] = 1;
   for(i=0; i<n; i++)
   {
      if(adj[v][i] == 1 && visited[i] == 0)
      {
         DFS(i);
      }
   }
}
int main() {
   int i, j, start;
   printf("Enter number of vertices : ");
   scanf("%d", &n);
   printf("Enter adjacency matrix : \n");
   for(i=0; i<n; i++)
```

Output :

Enter number of vertices : 6

Enter adjacency matrix : 1 2 3 3 4 4

1 2 3 4 5 6

8 4 5 6 7 8

4 5 6 7 8 9

3 4 6 7 8 9

2 3 4 5 6 7 8

Enter starting vertex (0 to 5) : DFS traversal : 8

INSTITUTE OF ENGINEERING AND TECHNOLOGY
(Mohanlal Sukhadia University, Udaipur)

Name..... Class..... Sem..... S. No.....

```
for (j=0; j<n; j++)
    scanf ("%d", &adj[i][j]);
for (i=0; i<n; i++)
    visited[i] = 0;
printf ("\n Enter starting vertex (0 to %d) :", n-1);
scanf ("%d", &start);
printf ("\n DFS traversal : ");
DFS (start);
return 0;
}
```

Experiment - 20

Aim :- WAP to show Basic operation of a Binary tree.

⇒
```
#include <stdio.h>
#define MAX 50
int tree[MAX];
int n=0;

/* Insert element */
void insert(int item) {
    if (n >= MAX-1) {
        printf("Tree is full!\n");
        return;
    }
    n++;
    tree[n] = item;
    printf("Item inserted successfully.\n");
}

/* Delete element */
void deleteItem(int item) {
    int i, found = 0;
    for (i=1; i<=n; i++) {
        if (tree[i] == item) {
            found = 1;
            break;
        }
    }
}
```

Output :

--- Array Based Binary tree ---

1. Insert
2. Delete
3. Inorder Traversal
4. Preorder Traversal
5. Postorder Traversal
6. Exit.

Enter your choice : 1

Enter item to insert : 10

Item inserted successfully.

```
if (! found) {
    printf ("Item not found ! \n");
    return ;
}
tree [i] = tree [n];
n --;
printf ("Item deleted successfull. \n");
}
/* Inorder Traversal */
void Inorder (int i) {
  if (i > n)
    return ;
  Inorder (2 * i);
  printf ("%d", tree [i]);
  Inorder (2 * i + 1);
}
/* Preorder Traversal */
void preorder (int i) {
  if (i > n)
    return ;
  printf ("%d", tree [i]);
  preorder (2 * i);
  preorder (2 * i + 1);
}
/* Postorder Traversal */
void postorder (int i) {
  if (i > n)
    return ;
```

```
postorder (2*i);
postorder (2*i+1);
printf ("%d", tree[i]);
}
/* Main function */
int main() {
  int choice, item;
  while (1) {
  printf ("\n -- Array Based Binary Tree --\n");
  printf ("1. Insert\n");
  printf ("2. Delete\n");
  printf ("3. Inorder traversal\n");
  printf ("4. Preorder Traversal\n");
  printf ("5. Postorder traversal\n");
  printf ("6. Exit\n");
  printf ("Enter your choice");
  scanf ("%d", &choice);
  switch (choice) {
  case 1;
    printf ("Enter item to insert:");
    scanf ("%d", &item);
    Insert (item);
    break;
  Case 2:
    printf ("Enter item to delete:");
    scanf ("%d", &item);
    delete_Item (item);
    break;
```

```
case 3:
    printf("\n Inorder Traversal : ");
    Inorder(1);
    printf("\n");
    break;
case 4:
    printf("\n Preorder Traversal : ");
    preorder(1);
    printf("\n");
    break;
case 5:
    printf("\n Post order traversal : ");
    postorder(1);
    printf("\n");
    break;
case 6:
    return 0;
default:
    printf("\n Invalid choice ! \n");
}
}
}
```