Experiment :
Installation, Environment setup & Starting with 'C' Language.

Date
Page No.

1) Write a code to print "Hello World"

Sol"
```
#include <stdio.h>

int main() {
    printf("Hello World \n");
    return 0;
}
```

2) Write a 'C' Program to ~~find~~ add two numbers, take number from user.

Sol"2)
```
#include <stdio.h>
int main() {
    int a, b, sum;
    printf("Enter First number:");
    scanf("%d", a);
    printf("Enter Second number:");
    scanf("%d", b);
    sum = a+b
    printf("The sum of two numbers is:%d", sum);

    return 0;
}
```

DELTA®

3. WAP to that Prompts the user to enter their name & age.

Sol:

```
#include <stdio.h>

int main() {
    int name, age;
    printf("Enter your name:");
    scanf("%s", name)
    printf("Enter your age:");
    scanf("%d", age);
    printf("Enter your name is %s
        and you are %d years old.";
            \n, name, age");
    return 0;
}
```

Experiment : 2

OPERATORS

Date _____

Page No. _____

1.) WAP a 'C' program to calculate the area & perimeter of a rectangle based on its length & width.

Sol'n

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

        printf ("Enter the length:");
        scanf ("%f", length);
        printf ("Enter the width:");
        scanf ("%f", width);

    area = length * width
    peri = 2 * (length + width)
    printf ("The area of rectangle is:", area);
    printf ("The perimeter of rectangle is: %f", peri);

        return 0;
}
```

Experiment :

Date

Page No.

2. WAP a 'C' program to ~~calculate the~~ convert temperature from Celsius to Farenheit using the formula $F = (C * 9/5) + 32$.

```
#include <stdio.h>
    float main () {
        printf ("Enter temp. in Celsius: ");
        scanf ("%f", celsius);
        F = (celsius * 9/5) + 32;
        printf ("Given temperature in
                farenheit is : %f \n", F);

        return 0;
    }
```

Experiment :

Date

Page No.

Experiment 3.1

Q.) W.A.P. to check if the triangle is valid or not. If the validity is established, do check if the triangle is isoceles, equilateral, right angle or scalene, take sides of the triangle as input from user.

```
#include <stdio.h>
int main() {
    int l1, l2, l3;
    printf("Enter legnth 1 of triangle:");
    scanf("%d", &l1);
    printf("Enter second side of triangle:");
    scanf("%d", &l2);
    printf("Enter third side of triangle:");
    scanf("%d", &l3);

    if ((l1+l2)>l3 && (l2+l3)>l1 &&
                (l1+l3)>l2) {
        printf("It is a valid triangle\n");
    }
    else, {
        printf("It is not a valid triangle\n");
    }
```

Experiment :

Date

Page No.

```
if ((l1==l2) && (l2==l3) {
        printf("It is an equilateral triangle");
    }
else if ((l1*l1 + l2*l2 = l3*l3) || (l2*l2 + l3*l3 = l1*l1)
        || (l3*l3 + l1*l1 = l2*l2)) {
        printf("It is a right angle triangle");
    }
else if ((l1==l2) || (l2==l3) || (l3==l1)) {
        printf("It is an isoceles triangle");
    }
else {
        printf("It is a scalen triangle");
    }
    return 0;
}
```

DELTA®

Experiment :

Date

Page No.

3.) W.A.P. to check if three points (x_1, y_1), (x_2, y_2) and (x_3, y_3) are collinear or not.

```
#include <stdio.h>
int main() {
    int x1, y1, x2, y2, x3, y3;

    printf("Enter coordinates of Point 1 (x1, y1):");
    scanf("%d %d", &x1, &y1);

    printf("Enter coordinates of 2nd point (x2, y2):");
    scanf("%d %d", &x2, &y2);

    printf("Enter coordinates of 3rd point (x3, y3):");
    scanf("%d %d", &x3, &y3);

    int a;
    a = x1*(y2-y3) + x2*(y3-y1) + x3*(y1-y2);

    if (a==0) {
        printf("The points are collinear");
    }
    else {
        printf("The points are not collinear");
    }
    return 0;
}
```

DELTA®

Experiment :

Date

Page No.

2) WAP to input the BMI index of the person & print the BMI value as per the foll. ranges :→

```
#include <stdio.h>
    int main() {
        float w, h, b;
        printf("Enter weight in kg:", w);
        scanf("%f", &w);
        printf("Enter height in meters:");
        scanf("%f", &h);
        b = w/(h*h);
        printf("Your bmi is:%f \n", b);
        if (b<15) {
        printf("Category: starvation \n");
        }
    else if (b>=15.0 && b<=17.5) {
        printf("Category: Anoramic");
    else if (b>=17.6 && b<=18.5) {
        printf("Category: underweight");
        }
    else if (b>=18.6 && bmi<=24.9) {
        printf("Category: Ideal");
    else if (b>=30 && b<=39.9) {
        printf("Category: obese");
        }
```

DELTA®

Experiment :

Date

Page No.

```
else {
    printf("Category : unspecified");
    return 0;
}
```

Experiment :

Date

Page No.

Bitwise Operator

1.) WAP to apply OR, AND & NOT operator on bit level

```
#include <stdio.h>
    int main () {
        int a = 12;
        int b = 4;
    printf ("a & b = %d \n", a & b);
    printf ("a | b = %d \n", a | b);
    printf ("~a = %d \n", ~a);

      return 0;
    }
```

2.) WAP to apply left shift & right shift operators.

```
#include <stdio.h>
    int main () {
      int a = 4;
    printf ("a << 1 = %d \n", a << 1);
    printf ("a >> 1 = %d \n", a >> 1);

      return 0;
    }
```

DELTA®

OUTPUT :→

a & b = 8

a | b = 14

~a = -13

OUTPUT :→

a << 1 = 8

a >> 1 = 2

Experiment :

Date

Page No.

3.1

3) W.A.P to check if three points (x_1, y_1); (x_2, y_2) & (x_3, y_3) are collinear or not.

```
#include <stdio.h>

int main () {
    int x1, y1, x2, y2, x3, y3;
    printf ("Enter the x co-ordinate of
                the first point: ", );
    scanf("%d", &x1)
    printf("Enter y1: ");
    scanf("%d", &y1);
    printf("Enter x2: ");
    scanf("Enter x2
    scanf("%d", &x2);
    printf("Enter y2: ");
    scanf("%d", &y2);
    printf("Enter x3: ");
    scanf("%d", &x3);
    printf("Enter y3: ");
    scanf("%d", &y3);

    int m1, m2;

    m1 = (y2 - y1) / (x2 - x1);
```

DELTA®

Experiment :

Date

Page No.

$m_2 = (y_3 - y_2) / (x_3 - x_2);$

$m_1 == m_2 ? m_1 : m_2$

printf("They are collinear");

$m_1 = m_2$? printf("They are collineal") : printf("Not Collinear");

return 0;

DELTA

Experiment :

Date

Page No.

3.1

1) WAP To check if the triangle is valid or not. If the validity is established, do check if the triangle is isoceles, equilateral, right angle, or scalene. Take sides of the triangle as input from a user.

```
#include <stdio.h>

int main() {
    int a, b, c;
                    // taking sides.
    printf("Enter the legnth of side 1");
    scanf("%d", &a);

    printf("Enter the legnth of side 2:");
    scanf("%d", &b);

    printf("Enter the legnth of side 3: ");
    scanf("%d", &c);

    // checking validity.

    if ((a+b>c) && (a+c>b) && (b+c>a)) {

    printf("The triangle is valid \n");

    }
```

DELTA®

Experiment :

Date

Page No.

```
// Checking Type :→

if (a==b && b==c) {
    printf("It is an equilateral triangle \n");
}
    else if (a==b || b==c || a==c) {
        printf("It is an isoceles triangle \n");
} else
    else if((a*a + b*b==c*c) || (a*a + c*c = b*b)
                || (c*c + b*b = a*a))
{   printf("Triangle is a right angled triangle \n");

    }
        else {
            print("It is a scalen triangle");

else {
    printf("\n The triangle is invalid ");

    }
}
    return 0;
}
```

DELTA®

Experiment :

Date

Page No.

4 According to the Gregorian calendar. It was Monday on the date 01/01/01. If any year is input through the Keyboard write a program to find out what is the day on 1st Janvary of this year.

```
#include <stdio.h>

int main () {
    int year, i;
    long total_days = 0;
    int day_number;

printf("Enter the year (like 2025): ");
 scanf ("%d", & year);

if (year < 1) {

    printf ("invalid input \n");

    return 1;
}

for (i = 1; i < year; i++) {

if ((i % 4 == 0 && i % 100 == 0) || (i % 400 == 0))
{ total_days = total_days + 366;
```

Experiment :

Date

Page No.

```
    }
    else {
        total_days = total_days + 365;
    }
}

    day_number = total_days % 7;

printf("\n The day on 1st Jan, %d was"
                                , year);

switch (day_number){

    case 0:
            printf ("Monday");
        break;
    case 1:
            printf ("Tuesday");
        break;

    case 2:
            printf ("Wednesday");
        break;

    case 3:
            printf ("Thursday");
        break;
```

DELTA®

Experiment :

Date

Page No.

```
case 4:
        printf ("Friday");
    break;

case 5:
        printf ("saturday");
    break;

case 6:
        printf ("Sunday");
    break;

    return 0;

}
```

Experiment :

Date

Page No.

5) WAP using ternary operator, the user should input the legnth & breadth of a rectangle, one has to find out which rectangle has the highest perimeter. The minimum number of rectangles should be three.

```
#include <stdio.h>

int main () {
    int l1, b1, p1;
    int l2, b2, p2;
    int l3, b3, p3;

printf ("Enter legnth and breadth of
                1st reclangle : ");
scanf ("%d %d", &l1, &b1);
    p1 = 2*(l1+b1);

printf ("Enter legnth & breadth of
                2nd rectangle : ");
scanf ("%d %d", &l2, &b2);

p2 = 2*(l2+b2);

printf ("Enter legnth & breadth of 3rd
                triangle : ");
scanf ("%d %d" &l3, &b3);
```

DELTA®

Experiment :

Date

Page No.

```
p3 = 2 * (l3 + b3);

int max_perimeter;

max_perimeter = (p1 > p2) ?
                (p1 > p3) ? p1 : p3) :
                (p2 > p3) ? p2 : p3);

if (p1 == max_perimeter) {
    printf("The 1st rectangle has maximum
                perimeter.");
}
else if (p2 == max_perimeter) {
    printf("The 2nd rectangle has maximum
                perimeter.")
}
else {
    printf("The 3rd rectangle has
                maximum perimeter");

return 0;
}
```

DELTA®

Experiment :

Date

Page No.

Ex. 3.2 :→ Loops.

Q.) W.A.P. to enter numbers till the user wants. At the end, it should display the count of positive, negative & zeroes entered.

```
#include <stdio.h>

int main() {
    int num;
    int pos_count = 0;
    int neg_count = 0;
    int zero_count = 0;
    char choice = 'y';

printf("Let's count the numbers :- \n");

while (choice == 'y' || choice == 'Y') {
    printf("Enter a number : ");

if (scanf("%d", &num) != 1) {

    printf("Invalid Input. \n");

        break;
}
```

DELTA

Experiment :

Date

Page No.

```
if (num>0) {
        pos_count ++;
}
   else if (num<0){
        neg_count ++;

   else {
        zero_count ++;
   }

   printf ("Do you want to enter another
                    number? (y/n): ");
   // clear input buffer.
   while (getchar() != '\n');

  scanf (" %c", &choice);

}

  printf ("Total no. of pos +ve integers:", pos_count);

printf("total no. of -ve integers :", neg_count);

printf ("Total no. of zeroes :", zero_count);

    return 0;
}
```

Experiment :

Date

Page No.

2.) WAP to print the multiplication table of the number entered by the user. It should be in the correct formatting.

```
#include <stdio.h>
    int num, i;

    printf("Enter the number for the multiplication
    scanf("%d", &num); table: ");

printf("\n Multiplication table for %d \n", num);

// Loop to generate table.

for (i=1; i<=10; i++) {

        int result = num* i;

        printf("%d * %d = %d \n", num, i, result);

}
        return 0;

}
```

DELTA®

Experiment :

Date

Page No.

3.) WAP to print the following output.

a)

```
1
2 3
4 5 6
```

```
#include <stdio.h>
int main () {
    int rows = 3;
    int i, j ;
    int count = 1;

for (i=1; i<=rows ; i++){

    for (j=1 ; j<=rows - i ; j++) {
        printf ("  ");
    }

    for (j=1 ;j <= i ; j++) {

    printf ("%d", count);
        count++ ;
    }
    print f ("\n");

}

    return 0;
}
```

DELTA®

Experiment :

Date

Page No.

b)

```
    1
  1 2 1
 2 3 3 1
1 4 6 4 1
```

```
#include <stdio.h>
int main(){
    int rows=5;
    int i, j ;
for(i=0; i<rows; i++){
    for(j=0; j<rows-i ; j++){
        printf(" ");
    }
int cur_value=1 ;
    for(j=0; j<=i; j++){
        if(j==0){
            cur_value=1;
        }
        else{
            cur_value= cur_value*(i-j+1)/j;
        }
    printf("%d", cur_value);
    }
    printf("\n");
}
return 0; }
```

DELTA

Experiment :

Date

Page No.

4. The population of a town is 1,00,000. The population has increased steadily at the rate of 10% per year for the last 10 years. Write a program to determine the population at the end of each year in the last decade.

```
#include <stdio.h>
    int main() {
        float current_pop = 100,000.0;
        float growth_rate = 0.10;
          int years = 10;
          int i;

    printf ("Year 10 :- 1,00,000.00 \n");

for (i=0; i>=1 ; i--) {

    current_pop = current_pop / (1.0 + growth_rate);

      printf (" Year %2d : %0.2f \n", i, current_pop);
}
    } return 0;

    }.
```

Experiment :

Date

Page No.

5.) Ramanujan no. is the smallest number that can be expressed as the sum of two cubes in two different ways. WAP to print all such numbers up to a reasonable limit.

Example of Ramanujan number: 1729.
$12^3 + 1^3$ & $10^3 + 9^3$ for a number
L = 20 (That is 'limit.)

```
#include <stdio.h>
int main () {
    int limits = 20;
    int a, b, c, d;

for (a=1; a<= limit ; a++) {
  for (b=a ; b<= limit ; b++){
    int n = a*a*a + b*b*b;

for for (c=a; c<=limit ; c++){
      for (d=c; d<=limit ; d++){
        int m = c*c*c + d*d*d;

if (n==m){

if (c>a || (c==a && d>b)){

printf ("%d = %d^3 + %d^3 = %d^3 + %d^3\n"
                                    ,n, a, b, c, d);
```

DELTA

Experiment :

Date

Page No.

Exp. 4.

1) Declare a global variable outside all functions & use it inside various functions to understand its accessibility.

```
#include <stdio.h>
  int global_counter=100;

void function_one(){
        global_counter=global_counter+6;

}

void func_two(){

    global_counter=global_counter+4;

}
   int main(){

      printf("Initial value:-%d\n", global_counter);

    function_one();
     func_two();

    printf("Final value:-%d\n", global_counter);
    return 0;
}
```

DELTA®

Experiment :

Date

Page No.

2.) Declare a local variable inside a function & try to access it outside the function. Compare this with accessing the global variable from the within the function.

```
#include <stdio.h>

    int global = 50; // Global Variable

void test() {

int local = 10; // Local variable.

  printf("Local variable value :- %d\n", local);
printf("Global variable value :- %d\n", global);

int main() {
      test();

printf("Global inside main :- %d\n", global);
printf("Attempt for local :- %d\n", local);
// The above line will not work as de 'main' func^n. doesn't know local.

  return 0;
}
```

DELTA

Experiment :

Date

Page No.

3.) Declare variables within different code blocks (enclosed by curly braces) & test their accessibility within & outside those blocks.

```
#include <stdio.h>
int main() {
        int outer_var = 10;
printf("1. Outer (main) scope : %d \n", outer_var);

{       // block start
        int inner_var = 20;
printf("2. Inside block : \n");
printf(" Access Outer : %d \n", outer_var);
printf(" Access Inner : %d \n", inner_var);
}
//block end (inner_var destroyed)
printf("3. Back in Outer scope. "

printf(" inner_var : %d \n", inner_var);

// Above line will show error.

        return 0;
}
```

DELTA

Experiment :

Date

Page No.

3.) Declare variables within different code blocks (enclosed by curly braces) & test their accessibility within & outside those blocks.

```
#include <stdio.h>
int main() {
    int outer_var = 10;
printf("1. Outer (main) scope : %d \n", outer_var);

{       // block start
    int inner_var = 20;
    printf("2. Inside block : \n");
    printf("Access Outer : %d \n", outer_var);
    printf("Access Inner : %d \n", inner_var);
}
//block end (inner_var destroyed)
printf("3. Back in Outer scope. "

printf("inner_var : %d \n", inner_var);

// Above line will show error.

    return 0;
}
```

DELTA®

Experiment :

Date

Page No.

4.) Declar a static local variable ~~that~~ inside a function. Observe how it's value persists accross function calls.

```
#include <stdio.h>

void inc_show(){

    static int count=1;

    printf("Call: %d, Value: %d \n", count, count);

    count++;
}

int main(){

    inc_show();
    inc_show();
    inc_show();

    return 0;
}
```

DELTA

Output :→

Call 1: Value is 1
Call 2: Value is 2
Call 3: Value is 3.

Experiment :

Date

Page No.

Exp:5

1-> WAP to read a list of integers & store it in a single dimensional array. Write a C program to print the second last integer in a list of integers.

```
#include <stdio.h>
    int main() {
        int n, i;

    printf("Enter the number of integers:");
     scanf("%d", &n);

    int arr[n]
     printf("Enter %d integers:\n", n);

      for(i=0; i<n; i++){

          scanf("%d", &arr[i]);

    int largest = arr[0];

    for(i=1; i<n; i++) {

        if (arr[i] > largest) {

            largest = arr[i];
        }
    }
```

DELTA®

Experiment :

Date

Page No.

```
int second_largest = arr[0];
for (i=0; i<n; i++){
    if (arr[i] < largest){
        second_largest = arr[i];
        break;
    }
}

if (second_largest == largest){
    printf("All numbers are the same no.
            No distinct second largest integer.\n");
    return 0;
}

for (i=0; i<n; i++){
    if (arr[i] > second_largest && arr[i] < largest){
        second_largest = arr[i];
    }
}

printf("The second largest integer is: %d\n",
                                second_largest);
return 0;
}
```

DELTA®

Experiment :

Date

Page No.

Q) WAP to read a list of integers & store it in a single dimensional array. Write a C program to count & display positive, negative, odd & even numbers in an array.

```
#include <stdio.h>
    int main() {
        int i, n;
        int a[50];
        int pos=0, neg=0, odd=0, even=0;

    printf("Enter the no. of elements : ");
     scanf("%d", &n);

    for (i=0; i<n; i++) {
            scanf("%d", &a[i]);
      }
       for(i=0; i<n; i++) {

          if (a[i]>0) {
                pos++;
          }
            else{
                neg++;
          }
            if (a[i]%2==0) {
                even++;
            }
```

DELTA®

:→

Experiment :

Date

Page No.

:→

```
else {
    odd ++;
}
for (i=0 ; i<n ; i++) {
    printf("%d %d %d %d", even, odd, pos, neg);
}
return 0;
}
```

3.) WAP to read a list of integers & store it in a single dimensional array. Write a C program to find the frequency of a particular number in list of integers.

```
#include <stdio.h>

int main (){
    int n;
    printf ("Enter the no. of elements: ");
    scanf ("%d", &n);

    int arr[n];

    printf ("Enter %d integers : \n", n);
```

DELTA

:→

Experiment :

Date

Page No.

:→

```
for (int i=0; i<n; i++){
        printf("Enter elements %d:", i++);
        scanf("%d", &arr[i]);
    if (~~i~~ ~~==~~ arr[i] == n)
    {
        count++;
    }
}
    printf("%d", count);
    return 0;
}
```

Q WAP that reads two matrices A(mxn) & B(pxq) & computes the product of A & B. Print both the input matrices & resultant matrix with suitable readings. Program must check the compatibility of orders of the matrices for multiplication

Sol"

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

DELTA

Experiment :

Date

Page No.

```
int m, n;
printf ("Enter the no. of rows & columns
        in 1st matrix :");
scanf ("%d %d", &m, &n);

int p, q;
printf ("Enter the no. of rows & columns :");
scanf ("%d %d", &p, &q);

int a[m][n], b[p][q];
printf("Enter 1st matrix elements :
for (int i=0; i<m; i++)
{
    for (int j=0, j<n; j++)
    {
        printf ("Enter 1st matrix element");

        printf ("Enter the a[%d][%d] element:"
                                    , i, j);
        scanf ("%d", &a[i][j]);
    }
}
    printf ("\n");

for (int i=0; i<p; i++) {
    for (int j=0; j<q; j++)
{
```

DELTA

→

Experiment :

Date

Page No.

```
printf("Enter the b[%d][%d] element of
            the second matrix:", i, j);
    scanf("%d", &b[i][j]);
    }
}

printf("MATRIX:->\n");
for(int i=0; i<m; i++){
  for(int j=0; j<n; j++){
    printf("%d\t", a[i][j]);
    }
    printf("\n");
  }
    printf("\n");
    printf("\n");

for(int i=0; i<p; i++){
    for(int j=0, j<q; j++){
      printf("%d\t", b[i][j]);
    }
    printf("\n");
}
```

DELTA®

Experiment :

Date

Page No.

```
if (n!= p) {

printf ("Matrix multiplication cannot
        be done");
        return 1;
}
    int c[m][q];

for (int i=0; i<m; i++) {
    for (int j=0; j<q; j++) {

            int sum = 0
    for (int k=0; k<n; k++)

sum = sum + (a[i][k] * b[k][j]);

        c[i][j] = sum;
    }
}
    printf ("Resultant product matrix \n");
    for (int i=0; i<m; i++) {
        for (int j=0; j<q; j++)
        {
            printf ("%d \t", c[i][j]);

        }
            printf ("\n");
    }
    return 0;
}
```

DELTA®

Experiment :

Date

Page No.

Exp. 6

1.) Develop a recursive & non-recursive func^n Fact (num) to find the factorial of a number, n! defined by fact (n) = 1, if n=0. Otherwise, fact (n) = n* Fact (n-1). Using this function, write a C programme to compute the binomial co-efficient. Tabulate the results for different value of n & r with suitable messages.

```
#include <stdio.h>

int fact (int n) {

    if (n == 0) {
        return 1;
    }
    else {
        return n* fact(n-1);
    }

int fact (int n)
{
    int product = 1;
for (int i = 2; i <= n ; i++) {
    product *= i;
}
    return product;
```

DELTA®

Experiment :

Date

Page No.

```
int ncr (int n; int r) {
    return fact(n)/(fact(n-r)* fact(r));
}
int main (){
    int n[5], r[5];
    for (int i=0; i<5; i++) {
    do{
        printf ("Enter value of n & r");
        scanf ("%d %d", & n[i], & r[i]);
    }
    while ((n[i] < 0 || (r[i] < 0) || (r[i] > n[i]))
}
    printf (" n  r  ncr \n");
    for (int i=0; i<5; i++ ){
        printf ("%d %d %d \n", n[i], r[i],
                                ncr(n[i], r[i]);
    }
    return 0;
}
```

DELTA®

Experiment :

Date

Page No.

3) Develop a recursive function GCD (num1, num2) that accepts two integer arguments. Write a C program that invokes with the function to find the GCD.

```
#include <stdio.h>
int gcd (int a, int b)
{
    if ((b==0) || (a<0)) {
        return a;
    }
    else
    {   return gcd(b, a%b);
    }
}
int main()
{
    int a, b;
    printf("Enter numbers");
    scanf("%d %d", &a, &b);

    c = gcd (a, b);
    printf("%d", c);

    return 0;
}
```

DELTA

Experiment :

Date

Page No.

2) Develop a recursive function GCD(num1, num2) that accepts two integer arguments. Write a C program that invokes with the function to find the GCD.

```
#include <stdio.h>
int gcd (int a, int b)
{
    if ((b==0)&&(a<0)) {
        return a;
    }
    else
    {   return gcd(b, a%b);
    }
}
int main()
{
    int a, b;
    printf("Enter numbers");
    scanf("%d %d", &a, &b);

    c = gcd(a, b);
    printf("%d", c);

    return 0;
}
```

DELTA®

Experiment :

Date

Page No.

3.) Develop a recursive func? FIBO(num) that accepts an integer argument. Write a C program that invokes this func? to generate the Fibonacci series upto num.

```
#include <stdio.h>

    int FIBO(int n)
    {
        if ((num == 1) || (num == 2))
    {
            return num-1;
        }
            else {
                return FIBO(n-1) + FIBO(n-2);
        }
}
        int main()
{
    int num;
    printf("Enter the number:");
    scanf("%d", &num);

  if (num <=0){
            printf("Invalid Input");

            return 1;
        }
```

DELTA®

Experiment :

Date

Page No.

```
for (int i=1 ; i<=n ; i++)
{
    printf("%d \t", FIBO(i));
}
    return 0;
}
```

4.) Develop a C function ISPRIME (num) that accepts an integer argument and returns 1, if argument is prime, a 0 otherwise. Write a C program that invokes this func? to generate prime numbers b/w the given range.

```
#include <stdio.h>

int ISPRIME (int num)
{
    int p=1;
    if (num == 1) {
        p=0;
    }
    else
for (int i=2; i<= num/2 ; i++) {
        if (num %i == 0) {
            p=0;
            break;
        }
    }
```

Experiment :

Date

Page No.

```
    return p;
}

int main ()
{
    int num1, num2;
    printf ("Enter the range :");
    scanf ("%d %d", &num1, &num2);

if ((num1<=0) || (num2<=0))
{
    printf ("Prime cannot be evaluated for
             non +ve no. ");
    return 0;
}
    if (num1 > num2)
{
    int temp = num1;
        num1 = num2;
        num2 = temp;
}
    for (int i = num1; i<=num2; i++)
{
        printf ("Prime cannot be evaluated
for       if (ISPRIME(i) == 1)
{
            printf ("%d \t", i);
}   }
    return 0;
}
```

DELTA®

Experiment :

Date

Page No.

Exp 8 :→ POINTERS.

1.) Declare different types of variables (int, float, char) & initialize them with the addresses of variables. Print values of both the pointers & variables they point to.

```
# include <stdio.h>
    int main()
{
    int a=5;
  float b=2.3;
   char c='?';
    int *d = &a;
     float *e = &b;
     char *f = &c;

printf ("Values :→ elements :→ %d %f %c"
          , *d, *e, *f);

printf ("Addresses :→ %p %p %p", d, e, f);

    return 0;

};
```

DELTA®

Experiment :

Date

Page No.

Q.2) Perform pointer arithmetic (increment & decrement) on pointers of different datatypes. Observe how the memory addresses change & the effects on data access.

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

    int a=5;
    float b=2.3;
    char c='?';
    int *d=&a;
    int *e=&b;
    int *f=&c;

    printf("%u\n",d);
      d=d+2;
    printf("%u\n",d);
    printf("%u\n",e);
    printf("%u\n",++e);
      printf("%u\n",--f);

    return 0;
}
```

This clears that on increment or decrement of increment or decrement of int & float changes address by 4 or 8 on

DELTA®

Experiment :

Date

Page No.

increment or decreament of character there is a change of 1 only.

Q3.) Write a function that accepts pointers as parametres Pass variables by reference using pointer & modify their values within the function

```
#include <stdio.h>
void change (int *a, int *b)
{
  *a=18;
  *b=45;
}
int main ()
{
  int a=7, b=10;
  printf ("%d\n%d\n", a, b);
  change (&a, &b);
  printf ("%d\n%d", a, b);

  return 0;
}
```

Output

```
7
10
18
45
```

Experiment :

Date

Page No.

Experiment 9 → File Handling in C.

1) Write a program to create a new file & write text into it.

```
#include <stdio.h>
#include <stdlib.h> // For exit function.

int main () {

FILE *filepointer; /* Making a variable filepointer
                      which keeps the track of file*/

const char *filename = "first.txt"
const char *
   char sentence[F]
   filePointer = fopen (filename, "w");

fprintf (file pointer, "Hello, this is my sentence.\n");

fclose (filepointer);
printf ("File closed.\n");

   return 0;
```

DELTA®

Experiment :

Date

Page No.

2.) Open an existing file & read its content character by character, and then close the file.

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

int main () {

 FILE * filepointer;

 int character ; // To distinguish character & EOF.

 const char * filename = "char.txt"

 filepointer = fopen (filename, "w");

 fclose (filepointer);

 filepointer = fopen (filename, "r");

 printf ("Reading :- "%s \n", filename);

 while ((character = fgetc(filepointer)) != EOF){

    putc (character, stdout);

 }
    fclose (file pointer);
      return 0;
```

DELTA®

Experiment :

Date

Page No.

3.) Open a file, read its content Line by Line, and display each line on the console.

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

#define MAX 256

int main () {

FILE *filepointer;

char line [MAX];

const char *filename = "sentence.txt";

~~filepointer = fopen (filename, "w");~~

~~fclose (file pointer);~~

~~file pointer~~

const char *sentence = "Hello, first line *\n"
                       "second line *\n"
                       "Last line \n"

filepointer = fopen (filename, "w");
```

DELTA®

Experiment :

Date

Page No.

```
fprintf (filepointer, "%s\n", sentence);
    fclose (filepointer);
    printf ("File successfully closed.\n");
    return 0;
```

---

10. Exp. 10 → Dynamic memory allocation.

1) Write a program to create a simple linked list in C using pointer & structure

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

struct node {
    int data;
    struct node *next;
};

void print (struct *node c){
    printf ("Linked List contents: \n");
    while (c != NULL){
```

DELTA®

Experiment :

Date

Page No.

```
while (c != NULL) {
    printf ("[%d] -> ", c -> data);
        c = c -> next;
    }
    printf ("NULL \n");
}

void insert (struct node ** hd, int d) {

struct node* n = (struct node*) malloc (sizeof (struct node));

    n -> data = d;
    n -> next = (*hd); // Link new node to
                       // old head.
    (*hd) = n; // make head point to new node.


int main () {
    struct node* head = NULL;

printf ("Creating List :- \n");

    insert (&head, 3);
    insert (&head, 2);
    insert (&head, 1);

    print (head);
        return 0;
```

DELTA®

Experiment :

Date _____

Page No. _____

2. W.A.P. to insert item in middle of the linked list

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

struct node {
    int data;
    struct node *next;
};

void print(struct node *c) {
    printf("Linked List contents:-\n");
    while (c != NULL) {
        printf("[%d] ->", c->data);
        c = c -> next;
    }
    printf("NULL\n");
}

// hd= head ; d= data
void insert(struct node** hd, int d){

struct node *n = (struct node*) malloc (sizeof (struct node));

n -> data = d;
n-> next = (*hd); //Link new node to old head.
(*hd) = n; //make head point to new node.
}
```

DELTA®

Experiment :

Date

Page No.

```
// hd = head , k = key , nd = new data

void mid(struct node *hd, int k, int nd){

        // n is the node we insert.
    struct node* n = (struct node*) malloc
                        (sizeof(struct node));
    n -> data = nd;
        // The above function will add
    // after node containing value 'k'.

    struct node* c = hd;

    while( c != NULL && c -> data != k){

        c = c -> next;
    }
        // Check if key was found.
    if (c == NULL){
        printf("Error key(%d) not found.", key);
        free(n); // Free memory bcoz new node
        return ;        //can't be used.
    }

        n -> next = c -> next;
    // c points to the node after which
       we want to insert.
    // new node now points to the node after 'c'.
```

DELTA®

Experiment :

Date

Page No.

```
        // 'c' next points to new node 'n'.
        c -> next = n;

        printf("Successfully inserted %d after
                                %d-\n", nd, k);
    }

// Clean up memory.

void clean(struct node** hd) {
    struct node* c = *hd;
    struct node* next node;

    while (c != NULL) {

        nextnode = c -> next // save pointer to
                             // next node before
                             // cleaning.
        free(c); // Release memory for the
                 // current node.
        c = next node; // move to next node;
    }
    *hd = NULL;
}

    int main() {
        struct node* head = NULL;
        printf("Create a linked list :- \n");
```

DELTA®

```
insert (&head, 3);
insert (&head, 2);
insert(&head, 1);

print (head);

mid (head, 2, 99);
                    // List is now : 1 -> 2 -> 99 -> 3
print (head);

clean (&head);

return 0;
}
```

Exp. 12 :→ Preprocessor & Directives.

1.) Write a program to define some constant variable in preprocessor.

```
#include <stdio.h>

#define MAX  50 // Integer const.

#define TAX  0.05 // Float const.
```

Experiment :

Date

Page No.

```
#define NAME "simple" //string const.

int main () {

    int users;
    printf ("Enter no. of users :- ");
    scanf ("%d", users);

    if (users <= MAX) {
        printf ("Current users (%d)
                are below the limit (%d).\n"
                , users, MAX);

    else {
    printf ("No. of users exceed limit (%d) "MAX);
    }

    float price = 200.0;
    float finalprice = price * (1.0 + TAX);

    printf ("Original price : %.2f \n", price);
    printf ("Tax Rate (%) : %.2f \n", TAX);
    printf ("Final price : %.2f \n", finalprice);

    printf ("This program is part of %s project.\n"
                                    , NAME);

    return 0;
}
```

DELTA®

Experiment :

Date

Page No.

2) W.A.P. to define a function in derivatives.

```
#include <stdio.h>

#define SUM(a,b) ((a)+(b))
    // when compiler sees. SUM(x,y), it replaces
       it with ((x)+(y)).

int main(){

    ~~int value1~~ int v1 = 10;
                   int v2 = 5;

    int r = SUM(v1, v2)

    printf("Using the sum macro :- \n");
    printf("%d + %d = %d", v1, v2, r);

    return 0;
}
```

DELTA®

Experiment :

Date

Page No.

Exp. 13 :→ Macros in C

1) Write a program to ~~create a static library~~ & define multiple macro to perform arithmetic functions.

```
#include <stdio.h>

#define ADD(a,b) ((a)+(b))

#define SUB(a,b) ((a)-(b))

#define MUL(a,b) ((a)*(b))

#define DIV(a,b) ((a)/(b))

int main () {

  int n1, n2;

  printf ("Enter the first number :- ");
   scanf ("%d", n1);

  printf ("Enter the second number:- ");
    scanf ("%d", n2);
                                  H
// ADDITION
  printf ("%d + %d = %d", n1, n2, ADD (n1, n2));
```

DELTA®

Experiment :

Date

Page No.

```
//SUBTRACTION.
  printf("%d - %d = %d", n1, n2, SUB(n1,n2));

//MULTIPLICATION
printf("%d X %d = %d ", n1, n2, MUL(n1,n2));

  //DIVISION

if (n2 != 0) {
     printf("%d / %d = %d", n1, n2, DIV(n1,n2));
 }
  else{
     printf("Division cannot be performed.");
 }

  return 0;
}
```

X

Exp. 14 :→ Static Library in C

```
// SECTION 1 :→ HEADER FILE

  int add(int a, int b);
  int sub(int a, int b);
  int mul(int a, int b);
                          // Save this as
                          // arithmetic.h
```

DELTA®

Experiment :

Date

Page No.

// SECTION 2 :-> SOURCE FILE (actual code)

```
int add(int a, int b) {        // Save this
    return a+b;                // as arithmetic.c
}
int sub(int a, int b){
    return a-b;
}
int mul (int a, int b){
    return a*b;
}
int div(int a, int b){
    if (b==0) {
        printf ("Division cannot happen.\n");
        return 0;
    }
    else
        return a/b;
}
```

// SECTION 3 :-> APPLICATION FILE.

```
int main(){
    int n1, n2;

    printf ("Enter the numbers you want to perform:-");
    scanf ("%d %d", &n1, &n2);
    // Addition.
    printf ("%d + %d = %d", n1, n2, ADD(n1, n2));
```

DELTA®

Experiment :

Date _____

Page No. _____

```
// Subtraction.
printf("%d - %d = %d", n1, n2, SUB(n1, n2));

// Multiplication.
printf("%d x %d = %d", n1, n2, MUL(n1, n2));

// Division
printf("%d / %d = %d", n1, n2, DIV(n1, n2));

return 0;    // Save this as main.c.
}
```

#

2.) Write a program to use static library in other program.

// After saving 3 files.

```
gcc -c arithmetic.c
ar rcs libarith.a arithmetic.o
gcc main.c -L. -larith -o app_runner
./app_runner
```

Exp. 15. Shared Library in C

Same code will be used just difference in compilation.

such as :→

DELTA®

Experiment :

Date

Page No.

1.) gcc -c -fPIC arithmetic.c

2.) gcc -shared -o libarith.so arithmetic.o

3.) gcc main.c -L. -larith -o app_runner_s

4.) To tell the location :→
export LD_LIBRARY_PATH=$PWD

5.) Run :- ./app_runner_s.

Exp.) Structures :→

1.)

```
#include <stdio.h>

typedef struct {
        float r;
        float i;
} C;

  void read (C *n)
void write (C n)
C add (C n1, C n2);
C sub (C n1, C n2);

void read (C* n) {
    printf ("Enter real, imaginary parts :- ");
    scanf ("%f %f", &n->r, &n->i);
```

DELTA®

Experiment :

Date

Page No.

```
void write (C n){
    printf ("%.1f + %.1fi", n.r, n.i);
}

C add (C n1, C n2){
    C res;
    res.r = n1.r - n2.r;
    res.i = n1.i - n2.i;
    return res;
}

int main () {
  C c1, c2, sum, diff;          // c1 & c2 are complex
                                // numbers.
  printf ("\n C1:\n");
   read (&C1);
  printf ("\n C2:\n");
   read (&c2);

  sum = add (c1, c2);
  diff = sub (c1, c2);

  printf ("\n C1: "); write (c1);
  printf ("\n C2: "); write (c2);
  printf ("\n sum: "); write (sum);
  printf ("\n diff: "); write (diff);

    return 0;
}
```

DELTA®

Experiment :

Date

Page No.

2.

```
#include <stdio.h>

#define DA 0.52
# define COUNT 3 // demo count

typedef struct {
        char name[10];
        float basic;
        float gross;
} Emp;

void write (Emp e)
    printf ("%-10s: %.2f\n", e.name, e.gross);

int main () {
        Emp staff[COUNT];

for(int i=0; i<COUNT; i++) {
   printf ("Emp %d Name :\n", i+1);
   scanf ("%9s", staff[i].name);

printf ("Emp %d Basic Pay: \n", i+1);
scanf ("%f", &staff[i].basic);

staff[i].gross = staff[i].basic * (1.0 + DA);
```

DELTA®

Experiment :

Date

Page No.

```
printf ("\n ---RESULTS---\n");

for (int i=0; i<COUNT; i++){
        write (staff [i]);
  }
     return 0;
}
```

3.)

```
#include <stdio.h>

typedef struct {
        int book_id;
        char title [50];
        char author [50];
        float price;
} Book;


void print (Book b){
     printf ("Book Details:- ");
     printf ("ID : %d\n", b.id);
     printf ("Title: %s \n", b.title);
     printf ("Author: %s\n", b.author);
     printf ("Price: %.2f\n", b.price);
}
```

DELTA®

Experiment :

Date

Page No.

```
int main (){

  Book myBook;

  printf ("Enter book ID: ");
   scanf ("%d", &myBook.id);

  printf ("Enter Title (1word): ");
  scanf ("%49s", myBook.title);

  printf ("Enter Author name (1 word) :- ");
  scanf ("%49s", mybook.author);

  printf ("Enter Price:- ");
  scanf ("%f", &myBook.price);

  printf ("\n");

   print (myBook);

   return 0;
}
```

DELTA®

Page No.

4.)

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

#define MAX 100

typedef union {
        char name [MAX];
        char address [MAX];
        char address_hostel [MAX];
        char city [MAX];
        char state [MAX];
        char zip [MAX];
} addressdata;

int main () {
    addressdata mydata;

    printf ("--- ADDRESS DISPLAY---");

char fulladdress[]="456 Union Blvd, Apt. C, New
                    York, NY, 10001";

strcpy (myData.address, fulladdress);
```

Experiment :

Date

Page No.

```
printf ("\n Your Present Address : \n");
printf ("-> %s\n", myData.address);

strcpy (myData.name, "Bob");

printf ("\n Note : The home adress is now corrupted
        /overwritten : %s\n", myData.address);

return 0;

}
```

DELTA®