INDEX

S. No.	Date	Title	Page No.	Remarks
1	06/08/25	Experiment 1: Installation, Environment setup and starting with C language	06/01	
	(i)	Write c program to print "Helloworld"	01	
	(ii)	Write ac program to print the address in multiple lines (new line)	02	
	(iii)	Write a program that prompts the user to enter their name and age	03	
	(iv)	Write a C program to add two numbers, take number from user	04-05	
2.	08/08/25	Operators		
	(i)	WAP a C program to calculate the area and perimeter of a rectangle based on its length and width.	06-07	
	(ii)	WAP a C program to convert temperature from Celsius to fahrenheit using the formula: $f = (c * 9/5 + 32)$	08/—	

INDEX

S. No.	Date	Title	Page No.	Remarks
3.1	13/08/25	Conditional Statement (3.1)	09-15	
4	13/08/25	Bitwise Operator (1.1)	16-19	
5	27/08/25	Loops (Ex-3.2)	20-33	
6	10/09/25	Variable and Scope of variable	34-36	
7.				

INDEX

Expt. No : 01 Date : 06/08/25

Experiment 1: Installation, Environment setup and Starting with c language.

Aim : To install the C compiler, set up the programming environment, and write basic C programs.

Theory: C is a general-purpose programming language developed by dennis Ritchie in 1972 It is widely used for system and application software development. Before coding, install a C compiler (GCC, Turbo C) and an IDE (Code: Blocks, Dev-C++, Vs Code). Set up environment paths, verify installation using gcc -- version; and start programming by writing, compili, and running C programs.

Teacher's Signature :

Page No : _____

Expt. No : 01 Date : 06/08/25

Expeviment : 01 Inotallation, Envivment wetup and Starting with C language

Wvite a C pvogvam to pvint "Hello Wovld"

```
1  #include <Studio.h>
2
3  int main() {
4        //Pvintf Hello, Wovld! to the wcvcen
5     Pvintf("Hello, Wovld!\n");
6     vetuvn 0;
7     }
8
```

Output :

```
Hello World
```

Teacher's Signature : _____

Expt. No : _____ Date : _____

3. Write a program that prompts the user to enter their name and age.

Coding:

```
#include <stdio.h>

int main() {
    char age;

    // Prompt for name
    printf("Please enter your name:");
    fgets(name, sizeof(name), stdin); // safer than gets()

    // Prompt for age
    printf("Please enter your age:");
    scanf("%d", &age);
    // Output the result
    printf("Hello, %s You are %d years old.\n", name, age);

    return 0;
}
```

Output

```
Please enter your name: ankit
Please enter your age : 18
Hello, ankit
You are 18 years old.
```

Teacher's Signature : _____

Expt. No : Date :

4 Write a C program to add number, take number from user...

Coding :

```
1  #include <stdio.h>
2
3  int main() {
4      int num1, number2, sum;
5
6      // Input from user
7      Printf ("Enter first number:");
8      scanf ("%d", &num1);
9
10     printf("Enter second number:");
11     scanf ("%d", &num2);
12
13     // Adition
14     sum = num1 + num2;
15
16     // out result
17     Printf ("sum = %d\n", sum);
18
19     return 0;
20     }
21
```

Teacher's Signature :

Expt. No : _____ Date : _____

Out Put:

```
Enter first number : 60
Enter second number : 40
Sum = 100
```

Teacher's Signature : _____

Expt. No : _____ Date : _____

Experiment 2: Operators:

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

Coding:

```
1  include <stdio.h>
2
3  int main() {
4      float length, breadth, area, perimeter;
5
6      // Input
7      printf("the length of the rectangle: ");
8      scanf("%f", &length);
9
10     Printf("the breadth of the rectangle:");
11     scanf("%f", &breadth);
12
13     // calculations
14     area = length * breadth;
15     perimeter = 2 * (length + breadth);
16
17     // output
18     printf("Area of the rectangle = %.2f \n", area);
19     printf("Perimeter of the rectangle = %.2f \n", perimeter);
```

Teacher's Signature : _____

Expt. No : _____ Date : _____

```
20
21  return 0;
22  }
23
```

Output

```
the length of the rectangle: 15
the breadth of the rectangle: 20
Area of the rectangle = 300.00
Perimeter of the rectangle = 70.00
```

Teacher's Signature : _____

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

2 WAP a C program to convert temperature from celsius to fahrenheit. Using formula f= (c* 9/5)+32

Coding:

```
#include <stdio.h>
int main() {
    float celsius fahrenheit;

    // input
Printf("temperature in celsius:");
scanf("%f", &celsius);

    // Conversion formula
fahrenheit= (celsius* 9/5) +32;

    // output
printf("Temperature in fahrenheit =%.2f°f\n", fahrenheit);

  return 0;
}
```

Output

temperature in celsius : 36

Temperature in fahrenheit = 96.80°f

Teacher's Signature : .....

Expt. No : _____ Date : _____

Experiment 3.1 : Conditional Statement

Aim : To write C programs using conditional statement to check triangle validity and calculate BMI.

Theory : Conditional statements in C help in decision-making
Triangle validity : sum of any two sides > third side. BMI formula : Weight / (height x height). BMI categories <15 Starvation 15.1-17.5 Anorexic, 17.6-18.5 Underweigh, 18.6-24.9 Ideal, 25-25.9 Overweight, 30-39.9 Obese ≥40 Morbidly Obese.

BMI Calculation :-

formula : $BMI = \frac{\text{Weight (Kg)}}{\text{height (m)} \times \text{height (m)}}$

BMI Ranges
Starvation - <15
Anorexation 15.1-17.5
Underweight 17.6-18.5
Ideal - 18.6-24.9
overweight - 25-25.9
obese - 30-39.9

Teacher's Signature : _____

Expt. No : _____ Date : _____

Program 1: WAP to take check if the triangle is valid or not If the valid or not. If the validity is established, do check if the triange is isoscceles equilateral, right angle, or scalene. Take sides of the triangle do input from a user.

Algorithm:

1. Start program, 2. Take three sides input. 3. Check validity 4. If valid, determine type. 5. Display result. 6 End program.

Code:

```
1  # include <stdio.h>
2   include <math.h>
3
4  int main() {
5      float a, b, c;
6
7      // Taking input
8  printf("Enter side a:");
9  Scanf("%f", &a);
10 printf("Enter side b:");
11 Scanf("%f", &b);
```

Teacher's Signature : _____

Expt. No : Date :

```
12  printf("Enter side c :");
13   scanf ("%lf", &c);
14
15    // Check triangle validity
16   if (a+b > c && a+c > b && b+c > a) {
17     Printf ("The triangle is valid.\n");
18
19   // Check for Equilateral
20    if (a == b && b == c) {
21     printf ("It is an Equilateral triangle.\n");
22   }
23     // Check for Isosceles
24  else if(a==b || b==c || a==c) {
25        printf("It is an Isosceles triangle.\n");
26   }
27    // Check for Right-angled using Pythagoras theorm.
28    else if (fabs((a*a + b*b) - (c*c)) < 0.0001 ||
29          fabs ((a*a + c*c) - (b*b)) < 0.0001 ||
30          fabs ((b*b + c*c) - (a*a)) < 0.0001) {
31        Printf (" It is a Right-angled triangle.\n");
32     }
33    // If none of the above, it's Scalene
34     else {
35      Printf("It is a Scalene triangle.\n");
36      }
```

Teacher's Signature :

Expt. No : _____ Date : _____

```
37  } else {
38      printf("The triangle is not valid.\n");
39  }
40
41  return 0;
42  }
43
```

Output

```
Enter side a: 3 4 5
Enter side b: Enter side c: The triangle is valid.
It is a Right-angled triangle.
```

Teacher's Signature : _____

Expt. No : _____ Date : _____

Program 2: WAP to computer the BMI Index of the person and print the BMI values as per the following ranges. You can use the following formula to compute BMI = Weight (Kgs)/Height (Mtrs)* Height (Mtrs).

Algorithm:

1. Start program. 2. Take weight and height input. 3 Calculate BMI. 4. Classify BMI. 5 Display result. 6. End program.

Code:

```
#include <stdio.h>

int main() {
    float wight, height, bmi;

    // Input weight and height
    Printf ("Your weight in Kilograms:");
    Scanf ("%f", &height);

    Printf (" Your hight in meters:");
    Scanf ("%f", &height);
```

Teacher's Signature : _____

Expt. No : _____ Date : _____

```
12      // Calculate BMI
13      bmi = weight / (height * height);
14
15      // Display BMI
16      Printf ("\n Your BMI is: %.2f\n", bmi);
17
18      // Determine BMI Category
19      if (bmi < 18.5) {
20          Printf ("Category: Underweight\n");
21      } else if (bmi >= 18.5 && bmi < 24.9) {
22          Printf ("Category: Normal weight\n");
23      } else if (bmi >= 25 && bmi < 29.9) {
24          Printf ("Category: Overweight\n");
25      } else {
26          Printf ("Category: Obese\n");
27      }
28
29      return 0;
30  }
31
```

Teacher's Signature : _____

Expt. No : Date :

Output

```
Your weight in kilograms: 70
Your height in meters. = 1.829

Your BMI is 14.95
Category: Underweight
```

Teacher's Signature :

Expt. No : Date : 16

EXPERIMENT-11 BITWISE OPERATOR

1. Bitwise Operator (AND, OR, NOT)

Aim :- To perform bitwise OR, AND, and NOT operations on integers in C

Theory :- Bitwise AND (&): Compares each bit of two number. If both are 1, result is 1, else 0.

- Bitwise OR (|) : Compares each bit of two number if at least one bit is 1, result is 1.
- Bitwise NOT (~): flips all bits of a number.

Algorithm.

1. Start
2. Take two integer inputs from the user
3. perform bitwise AND, OR, and NOT Operations
4. Display the results.
5. Stop.

Teacher's Signature :

Expt. No :

Date : 17

Code:-

```
1  # include<stdio.h>
2
3  int main() {
4      int a, b;
5  Printf("Enter two integers:");
6  Scanf("%d"%d" &a, &b);
7  Printf("a & b= %d \n", a & b);
8  Bitwise AND
9      Printf("a|b= %d \n", a|b);
10 Bit wise OR
11     Printf("~a = %d \n", ~a);
12 Bit wise NOT of a
13     Printf("~b= %d \n", ~b);
14 Bitwise NOT b
15     return 0;
16 }
```

Output

```
Enter two integers: 10  5
a & b=0
a|b= 15
~a= -11
~b= -6
```

Teacher's Signature :

Expt. No : Date : 18

2. Bitwise Shift Operators (left Shift, Right Shift)

Aim:- To demonstrate the working of left shift (<<) and right shift (>>) operators in C.

Theory :- left Shift (a<<n) :- Shifts bits of a to the left by n positions. Each shift multiplies the number by 2.
Right Shift (a >>n) :- Shifts bits of a to the right by n positions. each shift divides the number by 2.

Algorithm :-

1. Start
2. Take an integer input from the user
3. Take shift count as input.
4. Perform left shift and right shift
5. Display results.
6. Stop.

Teacher's Signature :

Expt. No : Date :

Code :-

```
#include <stdio.h>
int main () {
printf ("Enter a number: ");
 scanf ("%d", &a);
Printf ("Enter number of shifts:");
Scanf ("%d", &n);

Printf (" a<<n = %d\n", a<<n);
left Shift
printf (" a>>n = %d\n", a >> n);
 Right Shift
   return 0;
 }
```

Out put

```
Enter a number : 10
Enter number of shifts : 2
a << n = 40
a >> n = 2
```

Teacher's Signature :

Expt. No :

Date :

Experiment-32 Loops

1 WAP to enter number till the user wants. At the end it should display the count of positive, negative, and zeroes entered.

Aim: Read numbers until user stops and display counts of positive, nagative and zeros.

Theory: Use a loop that repeats whils the user wants to continue; examine sign of each number and increment counters.

Teacher's Signature :

Page No :

Expt. No :

Date :

Code :

```
# include <stdio.h>

int main() {
   int n;
   chav ch='y';
   int pos=0; neg=0, zevo=0;
   While (ch=='y' || ch=='Y') {
         Pvintf (" Entev an integev:");
         if (Scanf("%d", &n)!=1) vetuvn 0;
         if (n>0) pos++;
        else if (n<0) neg++;
       else zevo++;
       pvintf (" Continue? (y/n):");
       Scanf(" %c", &ch);
}
   Pvintf (" Positive: %od\n Negative: %d\n zevo: %od\n", pos, neg, zevo
   vetuvn 0;
}
```

Teacher's Signature :

Output

```
Enter an integer : 3
Continue? (y/n): y
Enter an integer : 2
Continue? (y/n) : y
Enter an integer: 0
Continue? (y/n) : n
Positive : 2
Nagetive : 0
zero : 1
```

2 Print multiplication table of number (proper formatting)

Aim: Printf multiplication table of user-entered number.

Theory: Use for loop from 1..n (often 1..10) and print num * i = product.

Code:

```
#include<stdio.h>
int main() {
    int num, i, upto=10;
    Printf("Enter number:");
    Scanf("%d",&num);
    Printf("Multiplication table of %d:\n" num);
    for (i=1; i<=upto; i++) {
        printf("%d * %2d = %d\n", num, i, num*i);
    }
    return 0;
}
```

Expt. No : Date :

Out put:

```
Enter number: 4
Multiplication table of 4:
4* 1=4
4* 2=8
4*3=12
4*4=16
4*5=20
4*6=24
4*7=28
4*8=32
4*9=36
4*10=40
```

Teacher's Signature :

Page No :

Expt. No : Date :

3a Pattern : rows with sequential numbers (example shown : 1 / 23 / 456)

Aim : Generate triangular pattern of sequential integers.

Theory : Maintain a counter variable that increments while printing rows with increasing counts.

Code :

```
#include<stdio.h>
int main() {
    int rows = 3;
    int val = 1;
    for (int r=1; r<=rows; r++) {
        for (int c=1; c<=r; c++) {
            printf("%d", val++);
        }
        printf("\n");
    }
    return 0;
}
```

Teacher's Signature :

Page No :

Expt. No :

Date :

Output

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

Teacher's Signature :

Page No :

Expt. No :

Date :

36 Pascal's triangle (1; 11; 121; 1331; 14641)

Aim: Print Pascall's triangle vows.

Theory: Use combinatorial formula C(n,K) or build intuatively using previous vow.

Code:

```
# include <stdio.h>

long long comb (int n, int K){
    long long vce=1;
    for (int i=1; i<=K; i++){
      vcs = vcs * (n-K+i) /i;
    }
   vetuvn vcs;
}

int main () {
  int vows = 5;
  for(int n=0; n<vows; n++){
      for (int K=0; K<=n; K++){
        pvint ("%lld ", comb (n,K));
    }
      pvint ("\n");
    }
   vetuvn 0;
  }
```

Teacher's Signature :

Expt. No : _____ Page No : _____

Date : _____

Output

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

Teacher's Signature : _____

Page No :

Expt. No :

Date :

4 Population growth at 10% per year for last 10 years :

Aim: Compute population at end of each year for 10 years given 10% annual growth.

Theory: Each year population = previous_population*(1+rate)
Use float / double for accuracy.

Code:

```
#include <stdio.h>

int main (){
    double pop = 100000;
    double rate = 0.10;
    for (int years=1 ; year<=10; year++){
     pop = pop* (1+rate);
     printf ("End of year %2d : %.0f \n", year, pop);
   }
   return 0;
}
```

Teacher's Signature :

Page No :

Expt. No :

Date :

Output

End of year 1: 110000
End of year 2: 121000
End of year 3: 133100
End of year 4: 146410
End of year 5: 161051
End of year 6: 177156
End of year 7: 194872
End of year 8: 214359
End of year 9: 235795
End of year 10: 259374

Teacher's Signature :

Page No :

Expt. No :

Date :

5 Population. Ramanujan numbers

(numbers expressible as sum of two cubes in two different ways)

Aim: Print number <= limit that can be written as a^3+b^3 = c^3 + d^3 with two distinct pairs.

Theory: for given limit L (max base), compute all a^3 + b^3 pairs and track sums that appear in more than one distinct pair.

Code:

Teacher's Signature :

Expt. No : Page No :

Date :

Code :

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

typedef struct { int a,b;} Pair;

int main () {
    int L;
    Printf (" Enter max base (e.g., 20): ");
    if (Scanf ("%d", &L) != 1) return 0;
    int maxSum = (L)*(L)*(L) + (L)*(L)*(L);
    int maxPairs = (L+1)*(L+1);
    int * sumVals = malloc (sizeof (int)* maxPairs);
    Pair * Pairs = malloc (sizeof (Pair)* maxPairs);
    int idx = 0;
   for (int a=1; a<=L; a++) {
    for (int b=a; b<=L; b++) {
      sumVals [idx] = a*a*a + b*b*b;
      pairs [idx]. a= a; pairs [idx] .b=b;
      idx ++;
    }
}
for (int i=0; i< idx; i++) {
    for (int j=i+1; j<idx; j++) {
    if (sumVals [i] == sumVals [j]) {
```

Teacher's Signature :

Expt. No :

Date :

```
        printf ("%d = %d^3 + %d^3 = %d^3 + %d^3\n",
                sumVals[i], pairs[i].a, pairs[i].b, pairs[j].a
        pairs[j].b);

            }
        }
    }
    free(sumVals); free(pairs);
    return 0;
}
```

Out put:

```
Enter max base (e.g. 20): 19
1729 = 1^3 + 12^3 = 9^3 + 10^3
4104 = 2^3 + 16^3 = 9^3 + 15^3
```

Teacher's Signature :

Expt. No : Date :

Expeviment-4

Vaviable and Scope of Vaviable

1 Global Vaviable used inside Vavious fonction.

Aim: Declave a global vaviable and accesos Imadify it in moltiple functions

Theovy: Global vavibles have program-wide scope and lifetime; accessible anywhevc aftev declavation.

Code

```
include < stdio.h>
int g= 10;

Void show () { pvintf ("g= %od\n", g); }
Void inc () { g++; }
inT main(){
    Show ();
    inc ();
    show ();
    vctuvn 0;
}
```

Teacher's Signature :

Page No : _____

Expt. No : _____ Date : _____

Out put

g= 10
g=11

Teacher's Signature : _____

2 Local Vaviable inside function and tvy to acce\o6 outside

Aim: Demonstvate a local vaviable is not acc essible outside its function.

Thcovy: Local vaviables have block/Function scope and avc destvayed when function vetuvns.

Codc:

```
#include<stdio.h>

Void f(){
  int local=5;
  pvintf("local inside f: %d\n", local);
}
int main(){
   f();
   vetuvn 0;
}
```

Output

local inside f: 5

Expt. No : _____ Date : _____

3 Variables inside different code blocks

Aim: Declare variables inside {} block and test accessibility

Theory: Variables declared inside a block exist only in that block.

Code

```
#include <stdio.h>
int main () {
  {
  int x=100;
  printf("%d\n", x);
  }
return 0;
}
```

Output

100

Teacher's Signature : _____

Expt. No : Date :

4 Static local Vaviable inside function

Aim : Show Static local vaviable vetains value acvoss calls

Theovy: static local vaviables ave allocated once and keep value fov pvogvam lifetime code:

Code:

```
#include <stdio.h>

void show() {
    static int cnt = 0;
    cnt ++;
    printf("cnt = %d\n", cnt);
}
int main() {
    show(); show(); show();
    vetuvn 0;
}
```

Teacher's Signature :

Expt. No : _____ Date : _____

Out put

cnt = 1
cnt = 2
cnt = 3

Expt. No : _____ Date : _____

Experiment-5 Array

1 Second largest integer in a list

Aim: Read array and print second largest.

Theory: Scan array keeping track of largest and second largest (handle duplicates).

Code:

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

int main() {
  int n;
  Printf("Number of elements: ");
  Scanf("%d", &n);
 int arr[n] ;
for(int i=0; i<n; i++) { Printf("a[%d]:", i); Scanf("%d",&arr[
int max= INT_MIN, second = INT_MIN;
for(int i=0; i<n; i++) {
    if(arr[i] > max) {second = max; max=arr[i];}
   else if(arr[i] >second && arr[i] != max) second=a
```

Teacher's Signature : _____

```
}
if (second == INT_MIN) printf ("No second largest (all equal)");
else printf ("Second largest = %d \n", second);
return 0;
}
```

Out put

```
Number of elements: 3 5 8
a[0] : a[1] : a[2] : 3
Second largest = 5
```

Page No :

Expt. No :

Date :

2 Count positive, negative, odd and even numbers in array :

Aim: Count categories in the array.

Theory: Single pass checking sign and parity.

Code

```
#include<stdio.h>

int main() {
    int n; printf("n:"); scanf("%d",&n);
    int pos=0, neg=0, odd=0, even=0, x;
    for(int i=0; i<n; i++){
        scanf("%d",&x);
        if(x>0) pos++; else if(x<0) neg++;
        if(x%2==0) even++; else odd++;
    }
    printf("pos=%d neg=%d odd=%d even=%d\n",
           pos, neg, odd, even);
    return 0;
}
```

Teacher's Signature :

Expt. No : Date :

Output

n= 4 6 7 -1 5

pos =3 neg =1 odd= 3 even=1

Page No :

Expt. No : Date :

3 frequency of a particular number

Aim: Count how many times a target appears.

Theory: Scan and increment when equal.

Code:

```
#include <stdio.h>

int main() {
    int n; printf("n: "); scanf("%d", &n);
    int arr[n];
    for (int i=0; i<n ; i++) scanf("%d", &arr[i]);
    int target ; printf("Target :"); scanf("%d", &tar
    int cnt = 0;
    for (int i=0; i<n; i++) if (arr[i]=target) cnt++;
    printf("frequency of %d = %d \n", target, cnt);
    return 0;
}
```

Teacher's Signature :

Expt. No : Date :

Output

n: -1 0 6 7 8
Target : frequency of 0=0

Teacher's Signature :

Expt. No : Date :

4 Matrix multiplication (with compatibility check)

Aim: Read matrices A (m×n) and B (p×q), check if if n==p, compute product

Theory: Product exists if columns of A == rows of B ; resultant size m×q.

Code

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

int main () {
    int m, n, p, q;
    printf ("Enter m n:"); scanf ("%d %d", &m, &n);
    printf ("Enter p q: "); scanf ("%d%d", &p, &q);
    if (n!=p) { printf ("Incompatible for multiplication\n");
    return 0; }
    int A[m][n], B[p][q], C[m][q];
    printf ("Enter A elements: \n");
    for (int i=0; i<m; i++) for (int j=0; j<n; j++) scanf
                                ("%d" & A[i][j]);
    printf ("Enter B elements: \n");
    for (int i=0; i<p; i++) for (int j=0; j<q; j++)
      scanf ("%d", &B[i][j]);
```

Teacher's Signature :

Expt. No : Date :

```
for (int i=0; i<m; i++) for(int j=0; j<q; j++){
        C[i][j]= 0;
    for (int K=0; K<n; K++) C[i][j] += A[i][K]*B[K]
        [j];
}
printf (" Resultant matrix C: \n");
 for (int i=0; i<m; i++){
     for (int j=0; j<q; j++) printf ("%d",C[i][j]);
     printf (" \n");
 }
 return 0;
}
```

Output :

```
Enter m n: 1 2
Enter p q: 3 4
Incompatible for multiplication
```

Teacher's Signature :

Expt. No : _____ Date : _____

Experiment-6 functions

1 fACT vecuvsive & non-vecuvsive t binomial coeffitent table

Aim Implcment function ~~factovial~~ both vc cuvsively & itcvatively; uscit to computc binomial coefficients C(n,v)

Thcovy : factovial(n) = n* fantovial(n-1) binomial= n! \ (v!(n-v)!). Bc cavcful with sizch(usc long long).

Teacher's Signature : _____

Expt. No : Page No :

Date :

Code :

```
#include <stdio.h>

long long fact_rec (int n){
    if (n<=1) return 1;
    return n * fact_rec (n-1);
}

long long fact_iter (int n){
    long long f=1;
    for (int i=2; i<=n; i++) f*=i;
    return f;
}

int main() {
    int nMax = 10;
    printf("n v C(n,v)\n");
    for (int n=0; n<=nMax; n++){
        for (int v=0; v<=n; v++) {
            long long c= fact_iter (n)/(fact_iter(v)*
            fact_iter (n-v));
        printf ("%2d %2d %4d\n", n, v, c );
    }
}
return 0;
}
```

Teacher's Signature :

Expt. No : Date :

Out put

n	v	C(n,v)
0	0	1
1	0	1
1	1	1
2	0	1
2	1	2
2	2	1
3	0	1
3	1	3
3	2	3
3	3	1
4	0	1
4	1	4
4	2	6
4	3	4
4	4	1
5	0	1
5	1	5
5	2	10
5	3	10
5	4	5
5	5	1
6	0	1
6	1	6
6	2	15

Teacher's Signature :

Page No :

Expt. No : Date :

2 Recursive GCD (num1, num2)

Aim: To develop a recursive function GCD (num1, num2 to find the greatest common divisor of two integers.

Theory: The GCD (Greatest Common Divisor) of two numbers is the largest number that divides both without remainder.

Code:

```
# include <stdio.h>
int GCD (int a, int b) {
    if (b == 0)
        return a;
    else
        return a;
    else
        return GCD (b, a % b);
}

int main () {
    int num1, num2;
    printf ("Enter two number:");
    scanf ("%d%d", & num1, & num2);
```

Teacher's Signature :

Expt. No : Date :

```
    printf("GCD of %od and %d = %d\n", num1, num2,
        GCD(num1, num2));
    return 0;
}
```

Output:

```
Enter two number : 4 5
GCD of 4 and 5 = 1
```

Teacher's Signature :

3 fibonacci Sevies Using Recovsive function

Aim: To develop a vecovsive function fIBO(num) to genevate the fibonacci Sequence up to a given number.

Theovy:

The fibonacci Sequence is a Sevies of numbevs where each tevm is the Sovm of the two pvecceding tevms

Code:

```
#include <stdio.h>

int fibo(int n){
    if (n== 0)
      vetuvn 0;
   elSe if (n==1)
      vetuvn 1;
   elSe
      Vetuvn fibo(n-1)+ fibo(n-2);
}
int main(){
```

Teacher's Signature :

Page No :

Expt. No :

Date :

```
int main () {
   int n, i;
   printf ("Enter number of terms: ");
   scanf ("%d", &n);

   printf ("Fibonacci Series:");
   for (i=0; i<n; i++)
     printf("%d", fibo(i));
   return 0;
}
```

Out put

```
Enter number of terms: 5 6 7 8
fibonacci Series: 0 1 1 2 3
```

Teacher's Signature :

Expt. No : Page No :

Date :

4 Prime Number Using function

Aim: To develop a function ISPRIME(num) that checks whether a number is prime, and use it to generate prime number in a given range.

Theory:

- A Prime number has exactly two distinct divisors 1 and itself
- The function checks divisibility from 2 to $\sqrt{n}$ If divisible it's not prime.

Code:

```
# include <stdio.h>

int ISPRIME (int num){
    if (num <= 1)
        return 0;
for (int i= 2; i*j<= num ; i++){
    if(num % i==0)
        return 0;
}
return 1;
}
```

Teacher's Signature :

Page No :

Expt. No :

Date :

```
int main () {
    int low, high;
    printf ("Enter range : ");
    scanf (%d%d, &low, &high);

    printf ("Prime number between %d and and%d :",
      low, high);
    for (int i = low; i <= high; i++) {
      if (ISPRIME (i))
        printf ("%d", i);
    }
  return 0;
}
```

Output

```
Enter range : 5  6
Prime number between 5 and 6 :  5
```

Teacher's Signature :

Page No :

Expt. No :

Date :

5 Reverse of a String Using function

Aim: To develop a function REVERSE that accepts a string & prints its reverse.

Theory: A string is a sequence of characters terminated by a null character '\0'.
To reverse, swap the first and last characters repeatedly until the middle is reached.

Code:

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

void Reverse(char str[]){
  int i, len;
  char temp;
  len= strlen(str);
  for(i=0; i<len/2; i++){
    temp= str[i];
    str[i]= str[len-i-1];
    str[len-i-1]= temp;
  }
}
```

Teacher's Signature :

Expt. No :

Date :

```
int main () {
  char str [100];
  printf ("Enter a string:");
  gets (str);

  REVERSE (str);
  Printf (" Reversed string : %s \n", str);
  return 0;
}
```

Output

Enter a string: ankit

Reversed string: tikna

Teacher's Signature :

Expt. No : Date :

Experiment 8 Pointers

1 Declare different pointer type and initialize with address

Aim: To declare different types of pointers (int, float, char) and initialize them with the address of variables.

Theory: A pointer is a variable that stores the memory address of another variable.

Code:

```
#include <stdio.h>
int main (){
    int a= 10;
    float b= 5.5;
    char c= 'A';
    int *p1= &a;
    float *p2= &b;
    char *p3 = &c;
```

Teacher's Signature :

Expt. No : Page No :

Date :

```
printf ("Address of a = %p, value = %d \n", p1, *p1);
printf ("Address of b = %p, value = %.2f \n", p2, *p2);
printf ("Address of c = %p, value = %c \n", p3, *p3);

    return 0;
}
```

Teacher's Signature :

Page No :

Expt. No :

Date :

2 Pointer avithmetic on pointers of different types

Aim: To perform pointer avithmetic (increment and decrement) on pointers of different date type and observe the memory address changes.

Theory Pointer avithmetic means performing operations like increment (ptr++) or decrement (ptr--) on pointers.

Code:

```
#include <stdio.h>
int main() {
  int a= 10, *p1= &a;
  float b= 5.5, *p2 = &b;
  char c= 'A' , *p3= &c;

printf ("Original addresses: \n");
printf ("int: %p \n float: %p \n char: %p \n", p1, p2, p3);

p1++; p2++; p3++;

printf ("\n After increment: \n");
printf ("int = %p \n float: %p \n char: %p \n", p1, p2, p3);
```

Teacher's Signature :

```
p1--; p2--; p3--;

printf("\n After decrement:\n");
printf("int: %p\n float: %p\n char: %p\n", p1, p2, p3);

return 0;
}
```

Output:

```
Original addresses:
int : 0x7ffcf42d6894
float: 0x7ffcf42d6890
char: 0x7ffcf42d688f

After increment
int: 0x7ffcf42d6898
float 0x7ffcf42d6894
char 0x7ffcf42d6890

After decrement
int: 0x7ffcf42d6894
float: 0x7ffcf42d6890
char: 0x7ffcf42d688f
```

Page No :

Expt. No :

Date :

3 function that accepts pointer as parameters - pass by reference and modify values

Aim: To write a function that accepts pointers as parameters, pass variables by reference using pointer, and modify via their values within the function.

Theory:

When we pass variables by reference using pointer any changes made inside the function reflect in the actual variables.

Code

```
#include <stdio.h>
void modify(int *x, float *y){
  *x=*x+10
 *y= *y+2.5;
}

int main() {
    Int a=5
    float b=3.5;
```

Teacher's Signature :

Expt. No :

Date :

```
printf("Before function: a=%d, b=%0.2f\n", a, b);
modify(&a, &b);
Printf("After function: a = %d, b= %0.2f\n", a, b);

return 0;
}
```

Out put

Before function: a = 5, b = 3.50
After function: a = 15, b = 6.00

Teacher's Signature :

Expt. No :

Page No : 29

Date :

Experiment 1: Structures and Union

1. Complex Number Using Structures and functions.

Aim- To write a C program that uses structures and function to read, write, add, and subtract complex number.

Theory- A complex number has a real part and an imaginary part.
Using structures in C, we can group these two values together.
functions allow modular programming - each task (read, print, add, subtract) is written separately

Teacher's Signature :

Expt. No :

Page No :

Date :

Program 1: Complex Number Operations

```
# include <stdio.h>

struct complex {
    float real;
    float imag;
};

struct complex readComplex(){
    struct complex c;
    printf("Enter real part:");
    scanf("%f", &c.real);
    printf("Enter imaginary part:");
    scanf("%f", &c.imag);
    return c;
}

void writeComplex(struct complex c){
    printf("%.2f + %.2fi\n", c.real, c.imag);
}

struct complex subtract(struct complex a, struct complex b){
    struct complex res;
    res.real = a.real - b.real;
    res.imag = a.imag - b.imag;
    return res;
}
```

Teacher's Signature :

Page No :

Expt. No :

Date :

```
int main() {

    struct Complex c1, c2, sum, diff;

    printf("Enter first complex number:\n");
    c1 = readComplex();
    printf("Enter second complex number:\n");
    c2 = readComplex();

    printf("\nfirst complex number:");
    writeComplex(c1);

    printf("Second complex number:");
    writeComplex(c2);

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

    printf("\nSum=");
    writeComplex(sum);

    printf("Difference=");
    writeComplex(diff);

    return 0;
}
```

Teacher's Signature :

Output

```
Enter first complex number:
Enter real part: 23
Enter imaginary part: 65
Enter second complex number:
Enter real part: 68
Enter imaginary part: 57

first complex number: 23.00+65.00i
Second complex number: 68.00+ 57.00i

Sum = 91.00 + 122.00i
Difference = -45.00 + 8.00i
```

Expt. No :

Page No :

Date :

2. Monthly Pay of 100 Employees

Aim- To compute the monthly pay of 100 employees using structures.

Theory - A structure can store related data such as employee name and salary DA (Dearness Allowance) is calculated as 59% of basic pay.
Gross salary = basic pay + DA.

Program 2: Employe Salary Calculation

```
#include<stdio.h>

Struct Employee {
    char name[30];
    float basic_pay;
    float gross_sallary;
};
int main() {
    struct Employe emp[100];
  int i;

for(i=0; i<100; i++){
    printf("\nEnter employee %d name: ",i+1);
    Scanf("%s", emp[i].name);
```

Teacher's Signature :

```
printf("Enter basic pay: ");
scanf("%f", &emp[i].basic_pay);

float da = 0.52 * emp[i].basic_pay;
emp[i].gross_salary = emp[i].basic_pay + da;
}

printf("\n--- Employee Salary Details ---\n");
for(i=0; i<100; i++){
    printf("%s - Gross Salary: %.2f\n", emp[i].name, emp[i]
    .gross_salary);
}
return 0;
}
```

Output

Enter employee 1 name : John
Enter basic pay : 10000

Enter employee 2 name : Mary
Enter basic pay : 15000

Enter employee 3 name: Obaka
Enter basic pay : 20000

...

(continues up to 100 employees)

Teacher's Signature :

Expt. No :

Page No :

Date :

3 Book Structure With function Argument

Aim

To create a Book structure and pass it to a function.

Theory:

Structures can be passed to functions by value. This allows writing reusable functions such as one that prints book details.

Program3: Book Structure.

```
#include<stdio.h>

struct Book{
    int book_id;
    char title[50];
    char author[30];
    float price;
};

Void printBook(struct Book b){
    printf("\nBook ID: %d", b.book_id);
```

Teacher's Signature :

Expt. No :

Page No :

Date :

```
    printf("\nTitle: %s" b.title);
    printf("\nAuthor: %s", b.author);
    printf("\nPrice: %0.2f\n", b.price);

}
int main() {
        struct Book b;

    printf("Enter book ID: ");
    scanf("%d", &b.book_id);

    printf("Enter title: ");
    scanf("%s", b.title);

    printf("Enter author name: ");
    scanf("%s", b.author);

    printf("Enter price: ");
    scanf("%f" &b.price);

    print Book(b);
    return 0;
}
```

OUTPUT

Enter book ID: 101
Enter title: C-Programming
Enter author name: Dennis
Enter price: 450.50

Teacher's Signature :

Expt. No :

Page No :

Date :

4 Union of 6 Address Strings

Aim

To store multiple address fields using a union and display the present address.

Theory

A union stores only one value at a time and shares memory between members.
The last written value is the one that remains.

Program 4: Union for Address

```
#include<stdio.h>

union Address {
    char name[30];
    char home_address[50];
    char hostel_address[50];
    char city[20];
    char state[20];
    char zip[10];
};
```

Teacher's Signature :

Expt. No :

Date :

```
int main () {
    union Address a;
    printf ("Enter your present address:");
    scanf ("%s", a.home_address);
    printf ("\n Your address is: %s\n", a.home_address);
    return 0;
}
```

Output :

Enter your present address : Bageshwar Uttarakhand

Teacher's Signature :

Expt. No : Date :

Experiment:9- File Handling in C

1 Create a new file and write text into it

Aim :

To write a C program that creates a new file and writes user-entered text into it.

Theory:

file handling in C uses FILE pointers and functions like:

- fopen()- open or create a file
- fprintf()- write formatted data to a file
- fclose()- close the file

Modes:

- "w" → create/write.
- "r" → read.
- "a" → append.

Teacher's Signature :

Page No :

Expt. No :

Date :

Program 1 : Create a file & write text

Code

```
# include <stdio.h>

int main(){
    FILE *fp;
    char text[100];

    fp = fopen("output.txt", "w");

    if (fp == NULL) {
    printf("Unable to create file.\n");
    return 1;
}

printf("Enter text to store in file: ");
fgets(text, sizeof(text), stdin);

fprintf(fp, "%s", text);
fclose(fp);
printf("file created and text written successfully.\n");
    return 0;
}
```

Teacher's Signature :

Expt. No :

Date :

Output

Enter text to store in file : Hello world
file created and text written successfully.

Teacher's Signature :

Page No :

Expt. No :

Date :

2 Read an existing file character by character

Aim: To read a file's content 6 one character at a time.

Theory

Character - based reading u6cn:
- fgetc(FILE *fp) - reads a single character
- EOF marks end of file

Program 2: Read file character by character

Code

```
# include <stdio.h>

int main () {
    FILE *fp;
    char ch;

fp = fopen ("output.txt", "r");

if (fp == NULL) {
```

Teacher's Signature :

Expt. No :

Date :

```
        printf("file not found.\n");
    }

    printf("file content:\n");

    while ((ch=fgetc(fp)) != EOF) {
        putchar(ch);
    }

    fclose(fp);
    return 0;
}
```

Output:

```
file content:
Hello world
```

Teacher's Signature :

Expt. No: Date:

3 Read file line by line

Aim: To open a file, read it's content line-by-line, and display each line.

Theory: Line-by-line reading uses:

- fgets(char *buffer, int size, FILE *fp)
- Reads until newline or buffer is full.

Program 3: Read file line by line

```
#include<stdio.h>
int main(){
  FILE *fp;
  char= fopen("output.txt", "r");
  if (fp== NULL) {

      printf("file not found.\n");
      return 1;
  }
  printf("file content:\n");
  while fgets(line, sizeof(line), fp)) {
  }

fclose(fp);
```

Teacher's Signature:

Expt. No :

Date :

```
    return 0;
}
```

Output

```
file content
Hello world
Welcome to C programming
This is a text file.
```

Teacher's Signature :

Experiment 10- DYNAMIC MEMORY ALLOCATION

Aim : To write C program that creates a singly linked list using dynamic memory allocation.

Theory: Dynamic memory functions:

- malloc() - allocate memory
- free() - deallocate
- Linked list consists of nodes dynamically connected through pointers.

A node contains:

- Data
- Pointer to next node

Expt. No :

Date :

Program 1 : Create a linked list

Code :

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

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

int main() {
    struct Node *head = NULL, *temp, * newnode;
    int n, value;

    printf("How many nodes?");
    scanf("%d", &n);

for (int i=0; i<n; i++){
    newnode = (struct Node*) malloc (sizeof(struct Node));
    printf("Enter value: ");
    scanf("%d", &value);

newnode -> data = value;
newnode -> next = NULL;

if (head == NULL){
```

Teacher's Signature :

Expt. No :

Page No :

Date :

```
            head = newnode;
        } else {
            temp = head;
            while (temp -> next != NULL)
                temp=temp -> next;

            temp -> next = newnode;
        }
    }

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

    return 0;
}
```

Teacher's Signature :

Expt. No : _____ Page No : _____

Date : _____

Out put

How many nodes? 4
Enter value : 10
Enter value : 20
Enter value : 30
Enter value : 40

Linked list : 10 -> 20 -> 30 -> 40 -> NULL

Teacher's Signature : _____

Page No :

Expt. No :

Date :

2 Insert an item in the middle of the linked list

Aim: To insert a new node at a specified position in a linked list.

Theory

Insertion involves:
- Creating a new node
- Traversing to the (position-1)th node

Adjusting pointers

program 2: Insert in the middle

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

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

void insertAtPos(struct Node** head int pos,it
                 ){
    newnode->data = value;
```

Teacher's Signature :

```
if (pos==1) {
    newnode->next = *head;
    *head = newnode;
    return;
}

Struct Node *temp = *head;
for (int i=1; i<pos-1 && temp; i++)
    temp = temp->next
if (!temp){
    printf(" Position out of range.\n");
    free(newnode)
    return;
}

newnode->next = temp->next
temp->next = newnode;
}

printf("\n Enter position to insert: ");
scanf("%d", &pos);
printf("Enter value to insert: ");
scanf("%d", &value);
insertAtPos(&head, pos, value);
printf("\nUpdated Linked List:\n");
return 0;
}
```

Teacher's Signature :

Expt. No : 12

Date :

Experiment-12 Preprocessor and Directives in C

1 Define Constant Variable using preprocessor

Aim: To define constants using the C preprocessor

Theory: The C preprocessor handles.

- #define → defining constants and macros
- # include → importing libraries.

Constants created using #define are replaced at compile time.

Teacher's Signature :

Page No :

Expt. No : Date :

Program1: Constant using preprocessor

Code:

```
#include<stdio.h>
#define PI 3.14159
#define MAX 100

int main() {
    Printf("Value of PI: %0.5f\n", PI);
    Printf("Maximum value: %d\n", MAX);
    return 0;
}
```

Output

```
Value of PI : 3.14159
Maximum value: 100
```

Teacher's Signature :

Page No :

Expt. No :

Date :

2 Define a function using preprocessor macro

Aim: To define a function-like macro using the preprocessor

Theory:

Macros can behave like functions:
- faster (no function call overhead)
- No type checking
- Written using #define

Program 2: function-like macro

Code:

```
#include <stdio.h>

#define SQUARE(x) (x * x)

int main() {
    int n = 5;
    printf("Square = %d\n", SQUARE(n));
    return 0;
}
```

Teacher's Signature :

Output :

Square = 25

Expt. No : 13 Date :

Experiment 13 Macros in C

1 Multiple macros for arithmetic operation

Aim To define macros for arithmetic operations

Theory

Macro definitions can perform mathematical expressions

- No runtime overhead.
- Substitute during preprocessing.

Teacher's Signature :

Page No :

Expt. No :

Date :

Program: Arithmetic macros

Code

```
# 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 x= 20, y=4

    printf("Addition: %d\n", ADD(x,y));
    printf("Subtraction: %d\n" SUB (x,y));
    printf("Multiplication: %d\n", MUL (x,y));
    printf(" Division: %d \n", DIV (x,y));

    return 0;
}
```

Teacher's Signature :

Expt. No :

Date :

Output :

Add : 24
Subtvaction : 16
MIti plication : 80
Division : 5

Teacher's Signature :

Expt. No : 14 Date :

Experiment 14 Static library in C

AIM

1. To create a static library for performing arithmetic functions in C.
2. To write a program that uses this static library.

Theory:

A static library (also called archive file or .a file) is a collection of object files that are linked into a program at compile time.

Characteristics of static libraries

- The library is copied into the executable at compile time
- Executable size increases.
- No need for the library at runtime.

Teacher's Signature :

Page No :

Expt. No :

Date :

CODE

1 avithmetic.c (Libvary Code)

```
#include "avithmetic.h"
int add (int a, int b) {
    vetuvn a+b;
}

int sub (int a, int b) {
    vetuvn a-b;
}
int mul (int a, int b) {
    vetuvn a*b
}
int diu (int a, int b) {
    vetuvn b != 0 ? a/b : 0;
}
```

2 avithmetic.h (Libvary Headev)

```
#ifndef ARITHMETIC_H
#define ARITHMETIC_H

int add (int, int);
int sub (int, int);
```

Teacher's Signature :

Expt. No : Date :

```
int mul (int, int);
int divi (int, int);

#end if
```

3 main.c

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

int main () {
    int a= 12, b= 4;

    printf(" Addition: %d\n", add(a,b));
    printf("Subtraction : %d\n", sub (a,b));
    printf ("Multiplication : %d\n", mul (a,b));
    printf (" Division: %d\n", divi (a,b));

    return 0;
}
```

Output:

```
Add : 24
Sub : 16
Mul : 80
Div : 5
```

Teacher's Signature :

Expt. No :

Date :

Experiment : 15

Shared Library in C

1 avithmetic.c (Library Code)

Code

```
#include "avithmetic.h"

int add(int a, int b) { return a+b; }
int sub (int a, int b) { return a-b; }
int mul (int a, int b) { return a*b; }
int divi (int a, int b) { return b != 0 ? a/b : 0; }
```

Teacher's Signature :

Page No :

Expt. No :

Date :

2 avithmetic.h

Code:

```
#ifndef ARITHMETIC_H
## define ARITHMETIC_H

int add(int,int);
int sub (int, int);
int mul (int, int);
int divi(int, int);

## endif
```

Teacher's Signature :

Expt. No :

Date :

3 main.c

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

int main(){
    int a= 15 ,b=3;

  printf("Addition: %d\n",add(a,b));
  printf("Subtraction: %d\n",sub(a,b));
  printf("Multiplication: %d\n",multa(a,b));
  printf("Division: %d\n",divi(a,b));

  return 0;
}
```

Out put:

```
Addition :18
Subtraction :12
Multiplication :45
Division :5
```

Teacher's Signature :