BVRITHYDERABAD

College of Engineering for Women

The Temple for Women Empowerment & Human Values

Approved by AICTE & Affiliated to JNTUH, Hyderabad
Nizampet Road, Bachupally, Hyderabad–500090,Telangana,India.

PPS Lab Record	
Department	Basic Science and Humanities
Year/Semester	I B.Tech I Semester
Subject	Programming for Problem Solving Lab
Academic Year(Regulation)	2023-24(BH23)
Subject Code	CS108ES

VISHNU
UNIVERSAL LEARNING

Vision

To emerge as the best among the institutes of technology and research in the country dedicated to the cause of promoting quality technical education.

Mission

At BVRITH , we Strive to

- Achieve academic excellence through innovative learning practices.
- Enhance intellectual ability and technical competency for a successful career.
- Encourage research and innovation.
- Nurturestudentstowardsholisticdevelopmentwithemphasisonleadershipskills Life skills and human values.

SYLLABUS

S.No	PROGRAMS	Date
	Practice sessions: a.Write a simple program that prints the results of all the operators available in C (includingpre/postincrement,bitwiseand/or/not,etc.).Readrequiredoperand values from standard input.	
	b.Write a simple program that converts one given data type to another using auto conversion and casting. Take the values form standard input.	
	Simple numeric problems: a.Write a program to find the max and min from the three numbers.	
	b.Write the program for the simple, compound interest.	
	c.Write program that declares Class awarded for a given percentage of marks, where mark <40%= Failed, 40% to <60% = Second class, 60% to <70%=First class, >= 70% = Distinction. Read percentage from standard input.	
	d. Write a program that prints a multiplication table for a given number and the numberofrowsinthetable.Forexample,foranumber5androws=3,theoutput should be: 5 x1 =5 5 x2 =10 5 x3 =15	
	e.Write a program that shows the binary equivalent of a given positive number between 0 to 255.	
	Expression Evaluation: a.A building has10 floors with a floor heigh to f 3meters each.A ball I sdropped from the top of the building. Find the time taken by the ball to reach each floor. (Use the formula s = ut+(1/2)at^2 where u and a are the initial velocity in m/sec (= 0) and acceleration in m/sec^2 (= 9.8 m/s^2)).	

	b. Write a C program, which takes two integer operands and one operator from the user,performstheoperationandthenprintstheresult.(Considertheoperators+,-,*, /,% and use Switch Statement)	
	c.Write a program that finds if a given number is a prime number	
	d.Write a C program to find the sum of individual digits of a positive integer and test given number is palindrome	
	e. A Fibonacci sequence is defined as follows: the first and second terms in the sequenceare0and1.Subsequenttermsarefoundbyaddingtheprecedingtwoterms in the sequence. Write a C program to generate the first n terms of the sequence.	
	f.Write a C program to generate all the prime numbers between1 and n,where n is a value supplied by the user.	
	g.Write a C program to find the roots of a Quadratic equation.	
	h.Write a C program to calculate the following,where x is a fractional value. 1 – x / 2 +x^2/4-x^3/6	
	i.Write a C program to read in two numbers,x and n,and then compute the sum of this geometric progression: 1+x+x^2+x^3+ +x^n. For example: if n is 3 and x is5,thentheprogramcomputes 1+5+25+125	
	Arrays and Pointers and Functions: a.Writea C program to find the minimum, maximum and average in an array of integers.	
	b.Write a functions to compute mean,variance, Standard Deviation, sorting of n elements in single dimension array.	

	c.Write a C program that uses functions to perform the following: Addition of Two Matrices Multiplication of Two Matrices Transpose of a matrix with memory dynamically allocated for the new matrix as row and column counts may not be same.	
	d.Write C programs that use both recursive and non-recursive functions To find the factorial of a given integer. To find the GCD (greatest common divisor) of two given integers. To find x ^ n	
	e.Write a program for reading elements using pointer in to array and display the values using array.	
	f. Write a program for display values reverse order from array using pointer.	
	g.Write a program through pointer variable to sum of n elements from array.	
	Files: a.Write a C program to display the contents of a file to standard output device.	
	b.Write a C program which copies one file to another, replacing all lowercase characters with their uppercase equivalents.	
	c.Write a C program to count the number of times a character occurs in a text file.The file name and the character are supplied as command line arguments.	
	d.Write aC program that does the following: It should first create a binary file and store 10 integers, where the file name and 10 values are given in the command line.(hint:convert the strings using atoi function) Now the program asks for an index and a value from the user and the value at that index should be changed to then ew value in the file.(hint:use fseek function)The program should then read all 10 values and print them back.	
	e.Write a C program to merge two files in to a third file(i.e.,the contents of the first file followed by those of the second are put in the third file).	

	Strings: a.Writea C program to convert a Roman numeral ranging from I to L to its decimal equivalent.	
	b.Write a C program that converts a number ranging from 1 to50to Roman equivalent	
	c.Write a C program that uses functions to perform the following operations: To insert a sub-string in to a given main string from a given position. To delete n Characters from a given position in a given string.	
	d. Write a C program to determine if the given string is a palindrome or not (Spelled same in both directions with or without a meaning like madam, civic ,noon, abcba, etc.)	
	e.Write a C program that displays the position of a character ch in the strings or –1 if S doesn't contain ch.	
	f.Write a C program to count the lines,words and characters in a given text.	
	Miscellaneous: a. Write a menu driven C program that allows a user to enter n numbers and then choose between finding the smallest, largest, sum, or average. The menu and all the choices are to be functions. Use a switch statement to determine what action to take. Display an error message if an invalid choice is entered.	
	b.Write a C program to construct a pyramid of numbers as follows: 1 * 1 1 * 12 * * 23 22 * * 123 * ** 456 333 * ** 4 4 44 * * *	
	Sorting and Searching: a.Write a C program that uses non recursive function to search for a Key value in a given list of integers using linear search method.	

	b.Write a C program that uses non recursive function to search for a Key value in a given sorted list of integers using binary search method.	
	c.Write a C program that implements the Bubble sort method to sort a given list of integers in ascending order.	
	d.Write a C program that sorts the given array of integers using selection sort in descending order	
	e.Write a C program that sorts the given array of integers using insertion sort in ascending order	
	f.Write a C program that sorts a given array of names	

Programs beyond Syllabus.

1. Given an integer n, for every integer i <= n, the task is to print "FizzBuzz" if i is divisible by 3 and 5, "Fizz" if i is divisible by 3, "Buzz" if i is divisible by 5 or i (as a string) if none of the conditions are true.

2. Starting with any positive integer N, Collatz sequence is defined corresponding to n as the numbers formed by the following operations :

   If n is even, then n = n / 2.
   If n is odd, then n = 3*n + 1.
   Repeat above steps, until it becomes 1.

3. Program to check whether the number is strong or not.
   A number can be said as a strong number when the sum of the factorial of the individual digits is equal to the number.
   For example, 145 is a strong number.

4. Program to check whether the given number is the Perfect number .
   A perfect number is a positive integer that is equal to the sum of its positive divisors, excluding the number itself.

5. Write a program to print lower triangular matrix and upper triangular matrix of an Array.
   Lower Triangular Matrix :A square matrix is called lower triangular if all the entries above the main diagonal are zero.
   Upper Triangular Matrix:A square matrix is called upper triangular if all the entries below the main diagonal are zero.

6. This C program uses a recursive function to solve the Tower of Hanoi puzzle. The Tower of Hanoi is a mathematical puzzle that consists of three rods and a number of disks of different sizes, which can slide onto any rod. The puzzle begins with all disks stacked on one rod in decreasing order of size, and the task is to move the entire stack to another rod, obeying the following rules:

a. Rules of Tower of Hanoi Puzzle:
b. Only one disk can be moved at a time.
c. Each move consists of taking the topmost disk from one of the stacks and placing it on top of another stack or on an empty rod.
d. No disk may be placed on top of a disk that is smaller than it.

7. Write program to store details of 60 students with following members Name, roll number , 6 subjects marks of each student, display Student name ,roll number with their average marks.

8. Write a program to find out string is palindrome or not using pointers.

9. Write a program to implement linear search in array of numbers ,searching for first occurrence of prime number ,if not return false.

10. Write a program to take list of N Numbers ,separate even and odd numbers and put them in two appropriate file (even.txt and odd.txt).

Program Name:

Write a simple program that prints the results of all the operators available in C (includingpre/postincrement,bitwiseand/or/not,etc.).Readrequiredoperand values from standard input.

Source Code:

```
#include <stdio.h>
int main()
{
        int a = 25, b = 5;
        int num = 10;
        printf("Arithmetic operators\n");
        printf("a + b = %d\n", a + b);
        printf("a - b = %d\n", a - b);
        printf("a * b = %d\n", a * b);
        printf("a / b = %d\n", a / b);
        printf("a % b = %d\n", a % b);
        printf("+a = %d\n", +a);
        printf("-a = %d\n", -a);
        printf("a++ = %d\n", a++);
        printf("a-- = %d\n", a--);
        printf("Relational operators\n");
        printf("a < b : %d\n", a < b);
        printf("a > b : %d\n", a > b);
        printf("a <= b: %d\n", a <= b);
        printf("a >= b: %d\n", a >= b);
        printf("a == b: %d\n", a == b);
        printf("a != b : %d\n", a != b);
```

```
printf("Logical operators\n");
printf("a && b : %d\n", a && b);
printf("a || b : %d\n", a || b);
printf("!a: %d\n", !a);
printf("Bitwise operators\n");
printf("a & b: %d\n", a & b);
printf("a | b: %d\n", a | b);
printf("a ^ b: %d\n", a ^ b);
printf("~a: %d\n", ~a);
printf("a >> b: %d\n", a >> b);
printf("a << b: %d\n", a << b);
printf("a = b: %d\n", a = b);
printf("a += b: %d\n", a += b);
printf("a -= b: %d\n", a -= b);
printf("a *= b: %d\n", a *= b);
printf("a /= b: %d\n", a /= b);
printf("a %= b: %d\n", a %= b);
printf("a &= b: %d\n", a &= b);
printf("a |= b: %d\n)", a |= b);
printf("a >>= b: %d\n", a >> b);
printf("a <<= b: %d\n", a << b);
printf("Special operators\n");
printf("sizeof(num) = %d bytes\n", sizeof(num));
printf("Conditional operator\n");
(a > b) ? printf("%d",a):printf("%d",b);
return 0;
```

```
}
```

Output:

```
Arithmetic operators
a + b = 30
a - b = 20
a * b = 125
a / b = 5
a % b = 0
+a = 25
-a = -25
a++ = 25
a-- = 26
Relational operators
a < b : 0
a > b : 1
a <= b: 0
a >= b: 1
a == b: 0
a != b : 1
Logical operators
a && b : 1
a || b : 1
!a: 0
Bitwise operators
a & b: 1
a | b: 29
```

a ^ b: 28

~a: -26

a >> b: 0

a << b: 800

a = b: 5

a += b: 10

a -= b: 5

a *= b: 25

a /= b: 5

a %= b: 0

a &= b: 0

a |= b: 5

a >>= b: 0

a <<= b: 160

Special operators

sizeof(num) = 4 bytes

Conditional operator

5

Program Name:

Write a simple program that converts one given data type to another using auto conversion and casting. Take the values from standard input**.**

Source Code:

```
//implicit type conversion
#include <stdio.h>

int main()

{

        int x = 10;

        char y = 'a';

        x = x + y;

        float z = x + 1.0;

        printf("Implicit type conversion\n");

        printf("x = %d, z = %f", x, z);

        return 0;

}

//explicit type conversion

#include<stdio.h>

int main()

{

         double x = 1.2;

         int sum = (int)x + 1;

         printf("sum = %d", sum);

         return 0;

}
```

Output:

Implicit type conversion

x = 107, z = 108.000000

Explicit type conversion
sum = 2

Simple numeric problems:

Program Name:

a. Write a program to find the max and min from the three numbers. **Source**

Code:

```
#include <stdio.h>
void main()
{
        int n1, n2, n3;
        printf("Enter three numbers: ");
        scanf("%d %d %d", &n1, &n2, &n3);
        if (n1>n2)
        {
                if(n1>n3)
                printf("%d is the largest number.", n1);
                else
                printf("%d is the largest number.", n3);
        }
        else
        {
                if(n2>n3)
                printf("%d is the largest number.", n2);
                else
                printf("%d is the largest number.",n3);
        }
 }
```

Output:

Enter three numbers: 96 56 82

96 is the largest number.

Enter three numbers: 76 106 82

106 is the largest number.

Enter three numbers: 16

85 is the largest number.

Program Name:

b. Write the program for the simple, compound interest. **Source**

Code:

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

int main()
{
    float p, t, r, si, ci;
    printf("Enter principal amount (p): ");
    scanf("%f", &p);
    printf("Enter time in year (t): ");
    scanf("%f", &t);
    printf("Enter rate in percent (r): ");
    scanf("%f", &r);
    si = (p * t * r)/100.0;
    ci = p * (pow(1+r/100, t) - 1);

    printf("Simple Interest = %0.3f\n", si);
    printf("Compound Interest = %0.3f", ci);
    return 0;
}
```

Output:

```
Enter principal amount (p): 100
Enter time in year (t): 2
Enter rate in percent (r): 50
Simple Interest = 100.000
Compound Interest = 125.00
```

Program Name:

c. Write program that declares Class awarded for a given percentage of marks, where mark <40%= Failed, 40% to <60% = Second class, 60% to <70%=First class, >= 70% = Distinction. Read percentage from standard input.

Source Code:

```
#include<stdio.h>
int main()
{
    int marks;
    printf("\n Enter Marks between 0-100:");
    scanf("%d", & marks);
    if (marks > 100 || marks < 0)
    {
      printf("\n Your Input is out of Range");
    }
    else if (marks >= 70)
    {
      printf("\n You got Distinction");
    }
    else if (marks >= 60)
    {
      printf("\n You got First Class");
    }
    else if (marks >= 40)
    {
      printf("\n You got Second Class");
    }
    else
    {
      printf("\n You got Failed");
```

```
    }
    return 0;
}
```

Output:

Enter Marks between 0-100 :70

You got Distinction

Program name:

Write a program that prints a multiplication table for a given number and the number of rows in the table.

For example, for a number 5 and rows=3, the output should be:

5 x 1 = 5

5 x 2 = 10

5 x 3 = 15

Source code:

```
#include <stdio.h>
int main()
{
        int n,r;
        printf("Enter an integer: ");
        scanf("%d", &n);
        printf("Enter the number of rows: ");
        scanf("%d",&r);

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

Output:

Enter an integer: 5

Enter the number of rows: 3

5 x 1 = 5

5 x 2 = 10

5 x 3 = 15

Program Name:

Write a program that shows the binary equivalent of a given positive number between 0 to 255.

Source Code:

```
#include<stdio.h>
int main()
{
	int a[10],n,i;
	printf("Enter the number to convert: ");
	scanf("%d",&n);
	for(i=0;n>0;i++)
	{
		a[i]=n%2;
		n=n/2;
	}
	printf("\nBinary of Given Number is=");
	for(i=i-1;i>=0;i--)
	{
		printf("%d",a[i]);
	}
	return 0;
}
```

Output:

```
Enter the number to convert: 33
Binary of Given Number is=100001
```

Program name:

A building has 10 floors with a floor height of 3 meters each. A ball Is dropped from the top of the building. Find the time taken by the ball to reach each floor. (Use the formula s =ut+(1/2)at^2 where u and a are the initial velocity in m/sec (= 0) and acceleration in m/sec^2 (= 9.8 m/s^2)).

Source code:

```
#include <stdio.h>
#include <math.h>
void main()
{
   int u = 0,s;
   float a = 9.8,t;
   int floor_height = 3;
   int num_floors = 10;
   for (int i = 1; i <= num_floors; i++)
  {
      s = i * floor_height;
      double t = sqrt((2 * s) / a);
      printf("Time taken to reach floor %d: %.2f seconds\n", i, t);
   }
}
```

Output:

Time taken to reach floor 1: 0.78 seconds
Time taken to reach floor 2: 1.11 seconds
Time taken to reach floor 3: 1.36 seconds
Time taken to reach floor 4: 1.56 seconds
Time taken to reach floor 5: 1.75 seconds
Time taken to reach floor 6: 1.92 seconds
Time taken to reach floor 7: 2.07 seconds
Time taken to reach floor 8: 2.21 seconds
Time taken to reach floor 9: 2.35 seconds
Time taken to reach floor 10: 2.47 seconds

Program name:

Write a C program, which takes two integer operands and one operator from the user, performs the operation and then prints the result.(Consider the operators +,-,*,/,% and use Switch Statement)

Source code:

```
#include <stdio.h>
void main()
{
   int num1, num2;
   char operator;
   printf("Enter two integers: ");
   scanf("%d %d", &num1, &num2);
   printf("Enter an operator (+, -, *, /, %): ");
   scanf(" %c", &operator);
   switch (operator)
   {
      case '+':
         printf("Result: %d\n", num1 + num2);
         break;
      case '-':
         printf("Result: %d\n", num1 - num2);
         break;
      case '*':
         printf("Result: %d\n", num1 * num2);
         break;
      case '/':
         if (num2 != 0)
            printf("Result: %d\n", num1 / num2);
         else
            printf(“error\”);
         break;
      case '%':
```

```
        if (num2 != 0)
            printf("Result: %d\n", num1 % num2);
        else
            printf("Error!\n”);
        break;
    default:
        printf("Error! Invalid operator.\n");
  }

}
```

Output:

Enter two integers: 50 5

Enter an operator (+, -, *, /, %): /

Result: 10

Program name:

Write a program that finds if a given number is a prime number

Source Code:

```
#include <stdio.h>
int main()
{
  int n, i, flag = 0;
  printf("Enter a positive integer: ");
  scanf("%d", &n);
  if (n == 0 || n == 1)
    flag = 1;
  for (i = 2; i <= n / 2; ++i)
  {
    if (n % i == 0)
    {
      flag = 1;
      break;
      }
  }
 if (flag == 0)
    printf("%d is a prime number.", n);
 else
    printf("%d is not a prime number.", n);

 return 0;
}
```

Output:

Enter a positive integer: 4

4 is not a prime number.

Program Name:

d. Write a C program to find the sum of individual digits of a positive integer and test given number is palindrome

Source Code:

```
#include <stdio.h>
int isPalindrome(int n)
{
   int reversed = 0, original = n;
   while (n != 0)
   {
      int digit = n % 10;
      reversed = reversed * 10 + digit;
      n /= 10;
   }
   return original == reversed;
}
int sumOfDigits(int n)
{
   int sum = 0;
   while (n != 0)
   {
      sum += n % 10;
      n /= 10;
   }
   return sum;
}
int main() {
   int num;
   printf("Enter a number: ");
   scanf("%d", &num);
```

```
    printf("Sum of digits: %d\n", sumOfDigits(num));
    if (isPalindrome(num))
    {
        printf("%d is a palindrome.\n", num);
    }
    else
    {
        printf("%d is not a palindrome.\n", num);
    }
    return 0;
}
```

Output:

Enter a number: 232

Sum of digits: 7

232 is a palindrome.

Program Name:

A Fibonacci sequence is defined as follows: the first and second terms in the sequence are 0 and 1.Subsequent terms are found by adding the preceding two terms in the sequence.

Write a C program to generate the first n terms of the sequence.

Source Code:

```
#include <stdio.h>
void main()
{
   int n, t1 = 0, t2 = 1, nextTerm;
   printf("Enter the number of terms: ");
   scanf("%d", &n);
   printf("Fibonacci Series: ");
   for (int i = 1; i <= n; ++i)
   {
      printf("%3d ", t1);
      nextTerm = t1 + t2;
      t1 = t2;
      t2 = nextTerm;
   }

}
```

Output:

Enter the number of terms: 5

Fibonacci Series: 0 1 1 2 3

Program Name:

Write a C program to generate all the prime numbers between1 and n, where n is a value supplied by the user.

Source Code:

```
#include<stdio.h>
void main(){
   int num,i,count,n;
   printf("Enter max range: ");
   scanf("%d",&n);
   printf("The prime numbers in the given range are: ");
   for(num = 2;num<n;num++)
   {
      count = 0;
      for(i=2;i<=num/2;i++)
      {
         if(num%i==0)
         {
            count=1;
            break;
         }
      }
      if(count==0)
         printf("%2d ",num);
   }
}
```

Output:

Enter max range: 7

The prime numbers in the given range are: 2 3 5

Program Name:

Write a C program to find the roots of a Quadratic equation **Source**

Code:

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

int main() {
   float a, b, c, discriminant, root1, root2;
   printf("Enter coefficients a, b and c: ");
   scanf("%f %f %f", &a, &b, &c);

   discriminant = b*b - 4*a*c;
   if (discriminant > 0) {
      root1 = (-b + sqrt(discriminant)) / (2*a);
      root2 = (-b - sqrt(discriminant)) / (2*a);
      printf("Roots are real and different.\n");
      printf("Root 1 = %.2f\n", root1);
      printf("Root 2 = %.2f\n", root2);
   }
   else if (discriminant == 0) {
      root1 = root2 = -b / (2*a);
      printf("Roots are real and the same.\n");
      printf("Root 1 = Root 2 = %.2f\n", root1);
   }
   else {
      float realPart = -b / (2*a);
      float imaginaryPart = sqrt(-discriminant) / (2*a);
      printf("Roots are complex and different.\n");
      printf("Root 1 = %.2f + %.2fi\n", realPart,imaginaryPart);
```

```
        printf("Root 2 = %.2f - %.2fi\n", realPart, imaginaryPart);
    }
    return 0;
}
```

Output:

Enter coefficients a, b and c: 1 2 1

Roots are real and the same.

Root 1 = Root 2 = -1.00

Program Name:

h. Write a C program to calculate the following, where x is a fractional value. 1 – x / 2 +x^2/4-x^3/6

Source Code:

```
#include <stdio.h>

int main()

{

        double x, result;

        printf("Enter the value of x: ");

        scanf("%lf", &x);

        result = 1 - x / 2 + (x * x) / 4 - (x * x * x)  6;
        printf("Result: %.4lf\n", result);

}
```

Output:

Enter the value of x: 2.5 Result: 1.9792

Program Name:

i. Write a C program to read in two numbers ,x and n, and then compute the sum of this geometric progression: 1+x+x^2+x^3+ +x^n. For example: if n is 3 and x is5, then the program computes 1+5+25+125

Source Code:

```
#include <stdio.h>
int main()
{
        int x, n;
        printf("Enter the value of x: ");
        scanf("%d", &x);
        printf("Enter the value of n: ");
        scanf("%d", &n);
        if (n < 0) {
                printf("Error: Please enter a non-negative integer for n.\n");

        }
        int sum = 0;
        int power = 1;
        for (int i = 0; i <= n; ++i) {
                sum += power;
                power *= x;
        }
        printf("Sum of the geometric progression: %d\n", sum);
}
```

Output:

Enter the value of x: 5 Enter the value of n: 3

Sum of the geometric progression: 156

Arrays and Pointers and Functions:

Program Name:

Write a C program to find the minimum, maximum and average in an array of integers.

Source Code:

```
#include <stdio.h>
int main()
{
	int n;
	printf("Enter the size of the array: ");
	scanf("%d", &n);
	if (n <= 0) {
		printf("Error: Please enter a valid positive size for the array.\n");
	}
	int arr[n];
	printf("Enter %d integers:\n", n);
	for (int i = 0; i < n; ++i) {
		scanf("%d", &arr[i]);
	}
	int min = arr[0];
	int max = arr[0];
	int sum = arr[0];
	for (int i = 1; i < n; ++i)
	{
		if (arr[i] < min)
		{
			min = arr[i];
```

```
            }
            if (arr[i] > max)
            {
                max = arr[i];
            }
            sum += arr[i];
        }
        double average = (double)sum / n;
        printf("Minimum: %d\n", min);
        printf("Maximum: %d\n", max);
        printf("Average: %.2f\n", average);
}
```

Output:

Enter the size of the array: 5

Enter 5 integers: 3 8 1 6 4

Minimum: 1 Maximum: 8 Average: 4.4

Program Name:

b. Write a functions to compute mean, variance, Standard Deviation, sorting of n elements in single dimension array.

Source Code:

```
#include <stdio.h>
#include<math.h>
void main () {
int i, n, sum;
float mean, varience, std_deviation;
printf ("enter number of values:");
scanf ("%d", &n);
int A[n];
printf ("Enter values of array:");
for (i = 0; i < n; i++)
   scanf ("%d", &A[i]);
for (i = 0; i < n; i++)
   mean= mean + A[i];
mean =mean/ n;
printf ("\n%f is the mean ", mean);
for (i = 0 ; i < n; i++)
   sum = sum + pow((A[i]-mean), 2);
varience=sum/n;
std_deviation = sqrt(varience);
printf("\n%f is varience\n%f is standard deviation", varience, std_deviation);
}
```

Output:

Enter the number of elements in the array: 5

Enter the elements of the array: 3 4 3 2 6

Sorted array: 2 3 3 4 6

Mean = 3.60

Variance = 1.84

Standard Deviation = 1.36

Program Name:

c. Write a C program that uses functions to perform the following: Addition of Two Matrices Multiplication of Two Matrices Transpose of a matrix with memory dynamically allocated for new matrix as row and column counts may not be same.

Source Code:

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

void add_matrices(int r, int c, int a[r][c], int b[r][c]) {
    int i, j;
    int sum[r][c];
    for(i = 0; i < r; i++) {
        for(j = 0; j < c; j++) {
            sum[i][j] = a[i][j] + b[i][j];
            printf("%d ", sum[i][j]);
        }
        printf("\n");
    }
}

void multiply_matrices(int r1, int c1, int a[r1][c1], int r2, int c2, int b[r2][c2]) {
    int i, j, k;
    if(c1 != r2) {
        printf("Multiplication not possible\n");
        return;
    }
    int product[r1][c2];
```

```
    for(i = 0; i < r1; i++) {
        for(j = 0; j < c2; j++) {
            product[i][j] = 0;
            for(k = 0; k < c1; k++) {
                product[i][j] += a[i][k] * b[k][j];
            }
            printf("%d ", product[i][j]);
        }
        printf("\n");
    }
}

void transpose_matrix(int r, int c, int a[r][c]) {
    int i, j;
    int **transpose = (int **)malloc(c * sizeof(int *));
    for(i = 0; i < c; i++) {
        transpose[i] = (int *)malloc(r * sizeof(int));
    }
    for(i = 0; i < r; i++) {
        for(j = 0; j < c; j++) {
            transpose[j][i] = a[i][j];
        }
    }
    for(i = 0; i < c; i++) {
        for(j = 0; j < r; j++) {
            printf("%d ", transpose[i][j]);
        }
        printf("\n");
    }
    for(i = 0; i < c; i++) {
        free(transpose[i]);
```

```
    }
    free(transpose);
}

int main() {
    int r1, c1, r2, c2, i, j;
    printf("Enter the number of rows and columns for the first matrix: ");
    scanf("%d %d", &r1, &c1);
    int a[r1][c1];
    printf("Enter the elements of the first matrix: ");
    for(i = 0; i < r1; i++) {
        for(j = 0; j < c1; j++) {
            scanf("%d", &a[i][j]);
        }
    }
    printf("Enter the number of rows and columns for the second matrix: ");
    scanf("%d %d", &r2, &c2);
    int b[r2][c2];
    printf("Enter the elements of the second matrix: ");
    for(i = 0; i < r2; i++) {
        for(j = 0; j < c2; j++) {
            scanf("%d", &b[i][j]);
        }
    }
    printf("Addition of matrices:\n");
    add_matrices(r1, c1, a, b);
    printf("Multiplication of matrices:\n");
    multiply_matrices(r1, c1, a, r2, c2, b);
    printf("Transpose of the first matrix:\n");
    transpose_matrix(r1, c1, a);
    return 0;
```

```
}
```

Output:

```
Enter the number of rows and columns for the first matrix: 3 3
Enter the elements of the first matrix: 1 2 3 4 5 6 7 8 9
Enter the number of rows and columns for the second matrix: 3 3
Enter the elements of the second matrix: 9 8 7 6 5 4 3 2 1
Addition of matrices:
10 10 10
10 10 10
10 10 10
Multiplication of matrices:
30 24 18
84 69 54
138 114 90
Transpose of the first matrix:
1 4 7
2 5 8
3 6 9
```

Program Name:

d. Write C programs that use both recursive and non-recursive functions To find the factorial of a given integer.

To find the GCD (greatest common divisor) of 2 given integers.

To find x ^ n

Source Code:

1.Factorial

```
#include <stdio.h>
// Recursive function
int factorial_recursive(int n) {
    if(n == 0)
        return 1;
    else
        return n * factorial_recursive(n - 1);
}
// Non-recursive function
int factorial_non_recursive(int n) {
    int fact = 1;
    for(int i = 1; i <= n; i++) {
        fact *= i;
    }
    return fact;
}
int main() {
    int num;
    printf("Enter a positive integer: ");
    scanf("%d", &num);
    printf("Factorial (Recursive): %d\n", factorial_recursive(num));
    printf("Factorial (Non-Recursive): %d\n", factorial_non_recursive(num));
```

```
    return 0;
}
```

Output:

Enter a positive integer: 4

Factorial (Recursive): 24

Factorial (Non-Recursive): 24

2. GCD

```
#include <stdio.h>
// Recursive function
int gcd_recursive(int a, int b) {
    if(b == 0)
        return a;
    else
        return gcd_recursive(b, a % b);
}
// Non-recursive function
int gcd_non_recursive(int a, int b) {
    while(b != 0) {
        int temp = b;
        b = a % b;
        a = temp;
    }
    return a;
}
int main() {
    int num1, num2;
    printf("Enter two positive integers: ");
    scanf("%d %d", &num1, &num2);
    printf("GCD (Recursive): %d\n", gcd_recursive(num1, num2));
    printf("GCD (Non-Recursive): %d\n", gcd_non_recursive(num1, num2));
    return 0;
}
```

Output:

```
Enter two positive integers: 6 3
GCD (Recursive): 3
GCD (Non-Recursive): 3
```

3.x^n

```
#include <stdio.h>
// Recursive function
int power_recursive(int base, int exp) {
    if(exp == 0)
        return 1;
    else
        return base * power_recursive(base, exp - 1);
}
// Non-recursive function
int power_non_recursive(int base, int exp) {
    int result = 1;
    for(int i = 0; i < exp; i++) {
        result *= base;
    }
    return result;
}
int main() {
    int base, exp;
    printf("Enter base and exponent: ");
    scanf("%d %d", &base, &exp);
    printf("Power (Recursive): %d\n", power_recursive(base, exp));
    printf("Power (Non-Recursive): %d\n", power_non_recursive(base, exp));
    return 0;
}
```

Output:

```
Enter base and exponent: 4 2
Power (Recursive): 16
Power (Non-Recursive): 16
```

Program Name:

e. Write a program for reading elements using pointer in to array and display the values using array.

Source Code:

```
#include<stdio.h>
void main()
{
   int a[100], i, *ptr,n;
   printf("enter the size of the array: ");
   scanf("%d",&n);
   printf("Enter the values of the array: ");
   for(i = 0; i < n; i++)
      scanf("%d", &a[i]);

   ptr = &a[n - 1];

   printf("\nElements of array in reverse order ...\n");
   for(i = 0; i < n; i++)
      printf("%d\n", *ptr--);
}
```

Output:

enter the size of the array: 4
Enter the values of the array: 1 2 3 4
Elements of array in reverse order ...4 3 2 1

Files:

Program Name:

a. Write a C program to display the contents of a file to standard output device.

Source Code:

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
        FILE *fp;
        char ch;
        fp=fopen("a.txt","r”);
        printf(“File contents are\n”);
        while((ch=getc(fp)!=’%’)
        {
            putchar(ch);
        }
        fclose(fp);
}
```

Output:

File contents are

Lohitha

Program Name:

b. Write a C program which copies one file to another, replacing all lowercase characters with their uppercase equivalents.

Source Code:

```
#include <stdio.h>
int main()
{
        FILE *sourceFile, *destinationFile;
        char sourceFilename[100], destinationFilename[100];
        char ch;
        printf("Enter the name of the source file: ");
        scanf("%s", sourceFilename);
        sourceFile = fopen(sourceFilename, "r");
        if (sourceFile == NULL)
        {
                printf("Could not open the source file %s.\n",
                sourceFilename);
        }
        printf("Enter the name of the destination file: ");
        scanf("%s", destinationFilename);
        destinationFile = fopen(destinationFilename, "w");
        if (destinationFile == NULL)
        {
                printf("Could not open the destination file %s.\n",
                destinationFilename); fclose(sourceFile);
```

```
    }
    while ((ch = fgetc(sourceFile)) != EOF)
    {
        if (ch >= 'a' && ch <= 'z')
        {
            ch = ch - 'a' + 'A';
        }
        fputc(ch, destinationFile);
    }
    fclose(sourceFile);
    fclose(destinationFile);
    printf("File copied successfully from %s to %s.\n", sourceFilename,
    destinationFilename);
}
```

Output:

```
Enter the name of the source file: source.txt
Enter the name of the destination file: destination.txt
File copied successfully from source.txt to destination.txt.
THIS IS A SAMPLE TEXT FILE.
IT CONTAINS SOME LOWERCASE CHARACTERS.
THIS PROGRAM WILL CONVERT THEM TO UPPERCASE.
```

Program Name:

c. Write a C program to count the number of times a character occurs in a text file.The file name and the character are supplied as command line arguments.

Source Code:

```
#include <stdio.h>
int main(int argc, char *argv[])
{
        if (argc != 3) {
                printf("Usage: %s <filename> <character>\n", argv[0]);
         }
         FILE *file = fopen(argv[1], "r");
         if (file == NULL) {
                printf("Error opening file: %s\n", argv[1]);
        }
         char targetChar = argv[2][0];
         int charCount = 0;
         char currentChar;
         while ((currentChar = fgetc(file)) != EOF)
         {
                (currentChar == targetChar)
                {
                        charCount++;
                }
         }
         fclose(file);
```

```
        printf("The character '%c' occurs %d times in the file %s\n", targetChar,
        charCount, argv[1]);
}
```

Output:

./count_char input.txt e

The character 'e' occurs 12 times in the file input.txt

Program Name:

e.Write a C program that does the following:

It should first create a binary file and store 10 integers, where the file name and 10 values are given in the command line.(hint: convert the strings using atoi function)

Now the program asks for an index and a value from the user and the value at that index should be changed to then ew value in the file.(hint:use fseek function)The program should then read all 10 values and print them back. The character are supplied as command line arguments.

Source Code:

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

int main(int argc, char *argv[])

{

        if (argc != 12) {

                printf("Usage: %s <filename> <num1> <num2> ... <num10>\n",
                argv[0]); return 1;

        }

        FILE *file = fopen(argv[1], "wb");

        if (file == NULL) {

                printf("Error creating file: %s\n", argv[1]);

        }

        for (int i = 2; i < argc; i++) {

                int num = atoi(argv[i]);
```

```
            fwrite(&num, sizeof(int), 1, file);
        }
        fclose(file);
    file = fopen(argv[1], "rb+");
        if (file == NULL) {
            printf("Error opening file: %s\n", argv[1]);
    }
        int index, newValue;
        printf("Enter the index (0-9) to update: ");
        scanf("%d", &index);
        if (index < 0 || index >= 10) {
            printf("Invalid index. Index should be between 0
            and 9.\n"); fclose(file);
                                        return 1;
        }
        printf("Enter the new value: ");
        scanf("%d", &newValue);
        fseek(file, index * sizeof(int), SEEK_SET);
        fwrite(&newValue, sizeof(int), 1, file);
        fseek(file, 0, SEEK_SET);
        printf("Updated values in the file:\n");
        int readValue;
        while (fread(&readValue, sizeof(int), 1, file) == 1) {
```

```
            printf("%d\n", readValue);
        }
        fclose(file);
}
```

Output:

./update_file data.bin 1 2 3 4 5 6 7 8 9 10

Enter the index (0-9) to update: 6

Enter the new value: 6

Updated values in the file:

1 2 3 4 5 6 6 8 9 10

Program Name:

e.Write a C program to merge two files in to a third file(i.e.,the contents of the first file followed by those of the second are put in the third file).

Source Code:

```
#include <stdio.h>

int main(int argc, char *argv[])

{

        if (argc != 4) {

                printf("Usage: %s <file1> <file2> <output_file>\n",
                argv[0]);

                return 1;

        }

        FILE *file1 = fopen(argv[1], "rb");

        if (file1 == NULL) {

                printf("Error opening file: %s\n", argv[1]);

        }

        FILE *file2 = fopen(argv[2], "rb");
        if (file2 == NULL) {

                printf("Error opening file: %s\n", argv[2]);

                fclose(file1);

        }

        FILE *outputFile = fopen(argv[3], "wb");

        if (outputFile == NULL) {
                printf("Error creating file: %s\n", argv[3]);
```

```
            fclose(file1);
            fclose(file2);
        }
        int ch;
        while ((ch = fgetc(file1)) != EOF) {
            fputc(ch, outputFile);
        }
        while ((ch = fgetc(file2)) != EOF) {
            fputc(ch, outputFile);
        }
        fclose(file1);
        fclose(file2);
        fclose(outputFile);
        printf("Files merged successfully into %s\n", argv[3]);
}
```

Output:

./merge_files file1.txt file2.txt merged_output.txt

Files merged successfully into merged_output.txt

Strings:

Program Name:

a. Write a C program to convert a Roman numeral ranging from I to L to its decimal Equivalent.

Source Code:

```
#include <stdio.h>
#include <string.h>
void main()
{
        int a[100],i,j,k,len;
        char roman[100];
        printf("Enter roman value\n");
        gets(roman);
        len = strlen(roman);
        for(i=0;i<len;i++)
        {
                if(roman[i] == 'I')
                        a[i] = 1;
                else if (roman[i] == 'V')
                        a[i] = 5;
                else if(roman[i] == 'X')
                        a[i] = 10;
                else if(roman[i] == 'L')
                        a[i] = 50;
                else if(roman[i] == 'C')
```

```
                a[i] = 100;
            else if(roman == 'D')
                a[i] = 500;
            else if(roman == 'M')
                a[i] = 1000;
            else
                printf("Invalid choice");
        }
        k = a[len-1];
        for(j = len-1;j>0;j--)
        {
            if(a[j]>a[j-1])
            k = k – a[j-1];
            else
            if(a[j] == a[j-1] || a[j] < a[j-1])
            k = k+a[j-1];
        }
        printf("%d",k);
}
```

Output:

Enter roman value

XVII 17

Program Name:

b. Write a C program that converts a number ranging from 1 to50 to Roman equivalent.

Source Code:

```
#include<stdio.h>
#include<stdlib.h>
void main()
{int num;
printf("enter the number:");
scanf("%d",&num);
while(num!=0)
        {if(num>50)
        {printf("not available");
        break;}
        else if (num==50)
        {printf("l");
         num =num- 50;}
         else if (num >= 40)
        {printf("xl");
           num -= 40;}
        else if (num >= 10)
        {printf("x");
           num -= 10;}
        else if (num >= 9)
        {printf("ix");
           num -= 9;}
        else if (num >= 5)
        {printf("v");
           num -= 5;}
        else if (num >= 4)
        {printf("iv");
           num -= 4;}

        else if (num >= 1)
        {printf("i");
           num -= 1;}
        else
          printf("not available");}
}
```

Output:

Enter a number between 1 and 50: 23

XXIII

Program Name:

c. Write a C program that uses functions to perform the following

operations: To insert a sub-string in to a given main string from a given position. To

delete n Characters from a given position in a given string.

Source Code:

```
#include <stdio.h>//deletion
#include<string.h>
void main() {
   char string1[20],string[40];
   int n,a,i,position,k=0;
   puts("\nenter the string:");
   gets(string1);
   a=strlen(string1);
   printf("\nenter the number of charecters to be deleted:");
   scanf("%d",&n);
   printf("enter the index from where it should get deleted: ");
   scanf("%d",&position);
   for(i=0;i<a;i++,k++)
   {    if(i==position)
         k=i+n;
      string[i]=string1[k];
   }
   string[i]='\0';
   printf("\nnewstring:%s",string);}
```

Output:

```
enter the string:happy new year
enter the number of charecters to be deleted:5
enter the index from where it should get deleted: 0
newstring: new year
```

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

```
#include<string.h>
void main() {
   char string1[20],string2[20],string[40];
   int n,a,b,i,j=0,k=0;
   puts("\nenter the string:");
   gets(string1);
   puts("\nenter the string to be inserted:");
   gets(string2);
   a=strlen(string1);
   b=strlen(string2);
   puts("\nenter the index where the word should be inseted:");
   scanf("%d",&n);
   for(i=0;i<=a+b-1;i++,k++)
   {    if(i==n)
          for(i=n;i<b;i++)
          {string[i]=string2[j];
          j++;}
      string[i]=string1[k];
   }
   string[i]='\0';
   printf("\nnewstring:%s",string);
```

Output:

```
enter the string:women
enter the string to be inserted:bvrit
enter the index where the word should be inseted:0
new string:bvritwomen
```

Program Name:

d. Write a C program to determine if the given string is a palindrome or not (Spelled same in both directions with or without a meaning like madam, civic ,noon, abcba, etc.)

Source Code:

```
#include <stdio.h>
#include<string.h>
int main()
{int i,j,l;
char string[10],rev[10];
printf("enter the string:");
gets(string);
l=strlen(string)-1;
for(i=0,j=l;i<=l;i++,j--)
        rev[j]=string[i];
if(strcmp(string,rev)==0)
   printf("palindrome");
else
   printf("not palindrome");
   return 0;
}
```

Output:

abbba is a palindrome

Program Name:

e. Write a C program that displays the position of a character ch in the strings or –1 if S doesn't contain ch.

Source Code:

```
//find the charecter
#include <stdio.h>
#include<string.h>
#include<stdlib.h>
int main()
{int i,l,result;
char string[10],charecter;
printf("enter the string:");
scanf("%s",string);
printf("enter the charecter to find:");
scanf(" %c",&charecter);
l=strlen(string)-1;
for(i=0;i<=l;i++)
      if(string[i]==charecter)
        { result=i;
              printf("\n%d",result);}
if(result==-1)
   printf("\n%d",result);
   return 0;
}
```

Output:

```
enter the string:student
enter the charecter to find:n
5
```

Program Name:

f. Write a C program to count the lines, words and characters in a given text.

Source Code:

```
//f)no.of lines,words,charecters
#include <stdio.h>
int main()
{char str[100];
int words=0,newline=0,characters=0;
printf("enter the text :");
scanf("%s",str);
for(int i=0;str[i]!='\0';i++)
 { if(str[i] == ' ')
    {words++;}
  else if(str[i] == '\n')
    {newline++;
    words++;}
  else if(str[i] != ' ' && str[i] != '\n')
    {characters++;}
  }
if(characters > 0)
    {words++;
    newline++;}
     printf("Total number of words : %d\n",words);
     printf("Total number of lines : %d\n",newline);
     printf("Total number of characters : %d\n",characters);
    return 0;
}
```

Output:

enter the text:this is pps record

Total number of words : 1

Total number of lines : 1

Total number of characters : 4

Miscellaneous:

Program Name:

a. Write a menu driven C program that allows a user to enter n numbers and then choose between finding the smallest, largest, sum, or average. Menu and the choices are to be functions. Use a switch statement to determine what action to take. Display an error message if an invalid choice is entered.

Source Code:

```
#include <stdio.h>
void inputNumbers(int n, int arr[]);
int findSmallest(int n, int arr[]);
int findLargest(int n, int arr[]);
int calculateSum(int n, int arr[]);
double calculateAverage(int n, int arr[]);
int main()
{
        int n;
        printf("Enter the number of elements: ");
         scanf("%d", &n);
        int numbers[n];
        inputNumbers(n, numbers);
        int choice;
        do {
                printf("\nMenu:\n");
```

```
printf("1. Find Smallest\n");
printf("2. Find Largest\n");
printf("3. Calculate Sum\n");
printf("4. Calculate Average\n");
printf("5. Exit\n");
printf("Enter your choice (1-5): ");
scanf("%d", &choice);
switch (choice) {
case 1:
        printf("Smallest number: %d\n", findSmallest(n, numbers)); break;
case 2:
        printf("Largest number: %d\n", findLargest(n, numbers));
         break;
case 3:
        printf("Sum: %d\n", calculateSum(n,
        numbers)); break;
case 4:
        printf("Average: %.2lf\n", calculateAverage(n,
        numbers)); break;
case 5:
        printf("Exiting program. Goodbye!\n");
        break;
default:
        printf("Invalid choice. Please enter a valid
option.\n");
```

```
            }
        } while (choice != 5);
}
void inputNumbers(int n, int arr[]) {
        printf("Enter %d numbers:\n", n);
        for (int i = 0; i < n; i++) {
            scanf("%d", &arr[i]);
        }
}
int findSmallest(int n, int arr[]) {
        int smallest = arr[0];
        for (int i = 1; i < n; i++) {
            if (arr[i] < smallest) {
                smallest = arr[i];
            }
        }
        return smallest;
}
int findLargest(int n, int arr[]) {
        int largest = arr[0];
        for (int i = 1; i < n; i++) {
            if (arr[i] > largest) {
                largest = arr[i];
```

```
                }
        }
        return largest;
}
numbers int calculateSum(int n, int arr[]) {
        int sum = 0;
        for (int i = 0; i < n; i++) {
        sum += arr[i];
        }
        return sum;
}
double calculateAverage(int n, int arr[]) {
        int sum = calculateSum(n, arr);
        return (double)sum / n;
}
```

Output:

Enter the number of elements:

512 5 8 19 3

Menu:

1. Find Smallest

2. Find Largest

3. Calculate Sum

4. Calculate Average

5. Exit

Enter your choice (1-5): 2

Largest number: 19

Enter your choice (1-5): 1

Smallest number: 3

Enter your choice (1-5): 3

Sum: 47

Enter your choice (1-5): 4

Average: 9.40

Enter your choice (1-5): 5

Exiting program. Goodbye!

Program Name:

b. Write a C program to construct a pyramid of numbers as follows:

```
1        *        1        1        *

12       * *      23       22       * *

123      * **     456      333      * **

4 4 44   * *

         *
```

Source Code:

```
//miscellaneous pyramids
#include<stdio.h>
void main()
{int n,i,j,k;
printf("enter number of rows:");
scanf("%d",&n);
for(i=1;i<=n;i++)
   {for(int j=1;j<=i;j++)
      printf("%d ",j);
   printf("\n");}
for (i=1;i<=n;i++ )
   {for(int j=1;j<=i;j++)
      printf("* ");
   printf("\n");}
for (i=1;i<=n;i++)
   {for(int j=1,k=1 ;j<=i;j++,k++)
      printf("%d ",k);
   printf("\n");}
for (i=1,k=1;i<=n;i++,k++)
   {for(int j=1;j<=i;j++)
      printf("%d ",k);
   printf("\n");}

for (i=1;i<=n;i++)
   {for(int j=1;j<= i;j++)
      printf("* ");
   printf("\n");}
for (i=n-1;i>=0;i--)
   {for(int j=1;j<=i;j++)
      printf("* ");
   printf("\n"); }

}
```

OUTPUT:

```
enter number of rows:3
1
1 2
1 2 3
*
* *
* * *
1
1 2
1 2 3
1
2 2
3 3 3
*
* *
* * *
* *
*
```

Sorting and Searching:

Program Name:

a. Write a C program that uses non recursive function to search for a Key value in a given list of integers using linear search method.

Source Code:

```
#include <stdio.h>

int linearSearch(int arr[], int n, int key) {

        for (int i = 0; i < n; i++) {
                if (arr[i] == key) {

                        return i;

                }

        }

        return -1;

}

int main() {

        int n, key;

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

        }

         printf("Enter the key to search: ");
```

```
        scanf("%d", &key);
        int result = linearSearch(arr, n, key);
        if (result != -1) {
                printf("Key %d found at index %d\n", key, result);
        }
        else {
                printf("Key %d not found in the list\n",
                key);
        }
}
```

Output:

Enter the number of elements in the list: 5 Enter the elements of the list: 10 23 7 14 18 Enter the key to search: 7

Key 7 found at index 2

Program Name:

b. Write a C program that uses non recursive function to search for a Key value in a given sorted list of integers using binary search method.

Source Code:

```
#include <stdio.h>

int binarySearch(int arr[], int low, int high, int key) {
        while (low <= high) {
                int mid = low + (high - low) / 2;
                if (arr[mid] == key) {
                return mid;
                }

                else if (arr[mid] < key) {
                        low = mid + 1;
                }
                else {
                high = mid - 1;
                }
        }
}
int main() {
        int n, key;
        printf("Enter the number of elements in the sorted list: "); scanf("%d", &n);
```

```
    int arr[n];
    printf("Enter the sorted elements of the list:\n");
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    printf("Enter the key to search: ");
    scanf("%d", &key);
    int result = binarySearch(arr, 0, n - 1, key);
    if (result != -1) {
        printf("Key %d found at index %d\n", key, result);
    }
    else {
        printf("Key %d not found in the list\n", key);
    }
}
```

Output:

Enter the number of elements in the sorted list: 8

Enter the sorted elements of the list: 2 5 8 12 16 23 38 45

Enter the key to search: 16

Key 16 found at index 4

Program Name:

c. Write a C program that implements the Bubble sort method to sort a given list of integers in ascending order.

Source Code:

```
#include <stdio.h>

void bubbleSort(int arr[], int n) {
        for (int i = 0; i < n - 1; i++) {
                for (int j = 0; j < n - i - 1; j++) {
                        if(arr[j] > arr[j + 1]) {
                        int temp = arr[j];
                        arr[j] = arr[j + 1];
                        arr[j + 1] = temp;
                        }
                }
        }
}
int main() {
        int n;
        printf("Enter the number of elements in the list: ");
        scanf("%d", &n); int arr[n];
        printf("Enter the elements of the list:\n");
        for (int i = 0; i < n; i++) {
                scanf("%d", &arr[i]);
```

```
        }
        bubbleSort(arr, n);
        printf("Sorted list in ascending order:\n"); for (int i = 0; i < n; i++) {
                printf("%d ", arr[i]);
        }
}
```

Output:

Enter the number of elements in the list: 6

Enter the elements of the list: 5 2 8 1 7 3

Sorted list in ascending order: 1 2 3 5 7 8

Program Name:

d. Write a C program that sorts the given array of integers using selection sort in descending order.

Source Code:

```
#include <stdio.h>

void selectionSortDescending(int arr[], int n) {
        for (int i = 0; i < n - 1; i++) {
                int maxIndex = i;
                for (int j = i + 1; j < n; j++) {
                        if (arr[j] > arr[maxIndex]) {
                                maxIndex = j;
                        }
                }
        if (maxIndex != i) {
                int temp = arr[i];
                arr[i] = arr[maxIndex];
                arr[maxIndex] = temp;
         }
        }
}
int main() {
int n;
```

```
printf("Enter the number of elements in the array: ");
scanf("%d", &n);

int arr[n];

printf("Enter the elements of the array:\n");

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

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

}

selectionSortDescending(arr, n);

printf("Sorted array in descending order:\n");

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

printf("%d ", arr[i]);

}

}
```

Output:

Enter the number of elements in the array: 7

Enter the elements of the array: 9 4 7 2 5 1 8

Sorted array in descending order: 9 8 7 5 4 2 1

Program Name:

e. Write a C program that sorts the given array of integers using insertion sort in ascending order.

Source code:

```
#include <stdio.h>

void insertionSortAscending(intarr[], int n) {
        for (int i = 1; i < n; i++) {
                int key = arr[i];
                int j = i - 1;
                while (j >= 0 && arr[j] > key) {
                        arr[j + 1] = arr[j];
                        j--;
                }
                arr[j + 1] = key;
        }
}

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

```
    }
    insertionSortAscending(arr, n);
    printf("Sorted array in ascending order:\n"); for
    (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
}
```

Output:

Enter the number of elements in the array: 6 Enter the elements of the array: 5 2 8 1 7 3 Sorted array in ascending order: 1 2 3 5 7 8

Program Name:

f. Write a C program that sorts a given array of names.

Source Code:

```
#include <stdio.h>

#include <string.h>

void bubbleSortStrings(char arr[][50], int n) {

        for (int i = 0; i < n - 1; i++) {

                for (int j = 0; j < n - i - 1; j++) {

                        if (strcmp(arr[j], arr[j + 1]) > 0) {

                                char temp[50];

                                strcpy(temp, arr[j]);

                                strcpy(arr[j], arr[j + 1]);

                                strcpy(arr[j + 1], temp);

                        }

                }

         }

}

int main() {

        int n;

         printf("Enter the number of names: ");

         scanf("%d", &n);

         char names[n][50];

         printf("Enter the names:\n");
```

```
    for (int i = 0; i < n; i++) {
        scanf("%s", names[i]);
    }
    bubbleSortStrings(names, n);
    printf("Sorted names in lexicographical order:\n");
    for (int i = 0; i < n; i++) {
        printf("%s\n", names[i]);
    }
}
```

Output:

Enter the number of names: 3

Enter the names: Lohitha Shruthi Pavani

Sorted names in lexicographical order:

Lohitha Shruthi Pavani

PROGRAMS BEYOND SYLLABUS

Program Name:

1. Given an integer n, for every integer i <= n, the task is to print “FizzBuzz” if i is divisible by 3 and 5, “Fizz” if i is divisible by 3, “Buzz” if i is divisible by 5 or i (as a string) if none of the conditions are true.

Source Code:

```
#include <stdio.h>

void main() {

        int n;

        printf("Enter a value for n: ");

        scanf("%d", &n);

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

                if (i % 3 == 0 && i % 5 == 0) {

                        printf("FizzBuzz\n");

                }

                else if (i % 3 == 0) {

                        printf("Fizz\n");

                }

                else if (i % 5 == 0) {

                printf("Buzz\n");

                }

                else {

                printf("%d\n", i);
```

```
            }
        }
}
```

Output:

```
Enter a value for n: 5
1
2
Fizz
4
Buzz
```

Program Name:

2. Starting with any positive integer N, Collatz sequence is defined corresponding to n as the numbers formed by the

following operations :

If n is even, then n = n / 2.

If n is odd, then n = 3*n + 1.

Repeat above steps, until it becomes 1.

Source Code:

```
#include <stdio.h>
void collatzSequence(int n) {
        while (n != 1) {
                printf("%d, ", n);
                if (n % 2 == 0) {
                        n = n / 2;
                }
                else {
                n = 3 * n + 1;
                }
        }
        printf("1\n");
}
int main() {
        int N;
        printf("Enter a positive integer N: ")
       scanf("%d", &N);
```

```
    if (N <= 0) {
        printf("Please enter a positive integer.\n");
    }
    printf("Collatz sequence for %d: ", N);
    collatzSequence(N);
}
```

Output:

Enter a positive integer N: 10

Collatz sequence for 10: 10, 5, 16, 8, 4, 2, 1

Program Name:

3. Program to check whether the number is strong or not.

A number can be said as a strong number when the sum of the factorial of the individual digits is equal to the number.

For example, 145 is a strong number.

Source Code:

```
#include <stdio.h>
int factorial(int num) {
        if (num == 0 || num == 1) {
                return 1;
        }
        else {
                return num * factorial(num - 1);
        }
}
int isStrongNumber(int num) {
        int originalNum = num;
        int sum = 0;
        while (num > 0) {
                int digit = num % 10;
                sum += factorial(digit);
                num /= 10;
        }
        return (sum == originalNum);
}
```

```
int main() {
        int num;
        printf("Enter a number: ");
        scanf("%d", &num);
        if (isStrongNumber(num)) {
                printf("%d is a strong number.\n", num);
        }
        else {
                printf("%d is not a strong number.\n", num);
        }
        return 0;
}
```

Output:

Enter a number: 145

145 is a strong number.

Program Name:

4. Program to check whether the given number is the Perfect number . A perfect number is a positive integer that is equal to the sum of its positive divisors, excluding the number itself.

Source Code:

```
#include <stdio.h>
int isPerfectNumber(int num) {
        int sum = 0;
        for (int i = 1; i <= num / 2; i++) {
                if (num % i == 0) {
                        sum += i;
                }
        }
        return (sum == num);
}
int main() {
       int num;
        printf("Enter a number: ");
        scanf("%d", &num);
        if (isPerfectNumber(num)) {
                printf("%d is a perfect number.\n", num);
        }
        else {
                printf("%d is not a perfect number.\n", num);
        }
}
```

Output:

Enter a number: 28

28 is a perfect number.

Program Name:

5. Write a program to print lower triangular matrix and upper triangular matrix of an Array. Lower Triangular Matrix :A square matrix is called lower triangular if all the entries above the main diagonal are zero.

Upper Triangular Matrix:A square matrix is called upper triangular if all the entries below the main diagonal are Zero.

Source Code:

```
#include <stdio.h>

void printLowerTriangularMatrix(int matrix[][100], int n) {

        printf("Lower Triangular Matrix:\n");

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

                for (int j = 0; j < n; j++) {

                        if (j > i) {

                                printf("0\t");

                        }

                        else {
                        printf("%d\t", matrix[i][j]);

                        }

                }

                printf("\n");

        }

}

void printUpperTriangularMatrix(int matrix[][100], int n)
```

```
{
    printf("Upper Triangular Matrix:\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            if (j < i) {
                printf("0\t");
            }
            else {
                printf("%d\t", matrix[i][j]);
            }
        }
        printf("\n");
    }
}
int main() {
    int n;
    printf("Enter the size of the square matrix: ");
    scanf("%d", &n);
    int matrix[100][100];
    printf("Enter the elements of the matrix:\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            scanf("%d", &matrix[i][j]);
```

```
            }
        }
        printLowerTriangularMatrix(matrix, n);
        printUpperTriangularMatrix(matrix, n);
}
```

Output:

Enter the size of the square matrix: 3

Enter the elements of the matrix:

1 2 3

4 5 6

7 8 9

Lower Triangular Matrix

1 0 0

4 5 0

7 8 9

Upper Triangular Matrix:

1 2 3

0 5 6

0 0 9

Program Name:

6. This C program uses a recursive function to solve the Tower of Hanoi puzzle. The Tower

of Hanoi is a mathematical puzzle that consists of three rods and a number of disks of different sizes, which can slide onto any rod. The puzzle begins with all disks stacked on one rod in decreasing order of size, and the task is to move the entire stack to another rod, obeying the following rules:

a. Rules of Tower of Hanoi Puzzle:

b. Only one disk can be moved at a time.

c. Each move consists of taking the topmost disk from one of the stacks and placing it on top of another stack or on an empty rod.

d. No disk may be placed on top of a disk that is smaller than it.

Source Code:

```
#include <stdio.h>

void towerOfHanoi(int n, char source, char auxiliary, char destination) {
        if (n == 1) {
                printf("Move disk 1 from %c to %c\n", source, destination);

                return;
        }

        towerOfHanoi(n - 1, source, destination, auxiliary);

        printf("Move disk %d from %c to %c\n", n, source, destination);
        towerOfHanoi(n - 1, auxiliary, source, destination);

}
int main() {

        int numDisks;

        printf("Enter the number of disks: ");
```

```
    scanf("%d", &numDisks);
    if (numDisks <= 0) {
        printf("Number of disks should be a positive integer.\n");
        }
    printf("Tower of Hanoi solution for %d disks:\n", numDisks);
    towerOfHanoi(numDisks, 'A', 'B', 'C');
    return 0;
}
```

Output:

Enter the number of disks: 3

Tower of Hanoi solution for 3 disks:

Move disk 1 from A to C

Move disk 2 from A to B

Move disk 1 from C to B

Move disk 3 from A to C

Move disk 1 from B to A

Move disk 2 from B to C

Move disk 1 from A to C

Program Name:

7. Write program to store details of 60 students with following members Name, roll number , 6 subjects marks of each student, display Student name ,roll number with their average marks.

Source Code:

```
#include <stdio.h>
#include <string.h>
struct Student {
        char  name[50];
        int rollNumber;
        float marks[6];
};
float calculateAverageMarks(float marks[], int numSubjects) {
        float sum = 0;
        for (int i = 0; i < numSubjects; i++) {
                sum += marks[i];
        }
        return sum / numSubjects;
}
int main() {
        const int numStudents = 60;
        const int numSubjects = 6;
        struct Student students[numStudents];
        for (int i = 0; i < numStudents; i++) {
                printf("Enter details for student %d:\n", i + 1);
                printf("Name: ");
                scanf("%s", students[i].name);
```

```
            printf("Roll Number: ");
            scanf("%d", &students[i].rollNumber);
            printf("Enter marks for 6 subjects:\n");
                for (int j = 0; j < numSubjects; j++) {
                    printf("Subject %d: ", j + 1);
                    scanf("%f", &students[i].marks[j]);
                }
            }
        printf("\nStudent Details with Average Marks:\n");
        for (int i = 0; i < numStudents; i++) {
            float averageMarks = calculateAverageMarks(students[i].marks,
            numSubjects);
            printf("Name: %s, Roll Number: %d, Average Marks: %.2f\n",
            students[i].name, students[i].rollNumber, averageMarks);
        }
}
```

Output:

```
Enter details for student 1:
Name: John
Roll Number: 101
Enter marks for 6 subjects:
Subject 1: 80
Subject 2: 75
Subject 3: 90
Subject 4: 85
Subject 5: 88
```

Subject 6: 92

Enter details for student 2:

Name: Alice

Roll Number: 102

Enter marks for 6 subjects:

Subject 1: 95

Subject 2: 87

Subject 3: 92

Subject 4: 78

Subject 5: 85
Subject 6: 90

... (input for 58 more students) ...

Student Details with Average Marks:

Name: John, Roll Number: 101, Average Marks: 86.67 Name: Alice,

Roll Number: 102, Average Marks: 88.17 ... (output for 58 more

students) ...

Program Name:

8. Write a program to find out string is palindrome or not using pointers.

Source Code:

```
#include <stdio.h>
#include <string.h>
int isPalindrome(char *str) {
        int length = strlen(str);
        char *start = str;
        char *end = str + length - 1;
        while (start < end) {
                if (*start != *end) {
                        return 0;
                }
        start++;
        end--;
        }
        return 1;
}
int main() {
        char inputString[100];
        printf("Enter a string: ");
        scanf("%s", inputString);
        if (isPalindrome(inputString)) {
                printf("%s is a palindrome.\n", inputString);
        }
        else {
```

```
            printf("%s is not a palindrome.\n", inputString);
        }
        return 0;
}
```

Output:

Enter a string: radar is a palindrome.

Program Name:

9. Write a program to implement linear search in array of numbers ,searching for first occurrence of prime number ,if not return false

Source Code:

```
#include <stdio.h>
int isPrime(int num) {
        if (num <= 1) {
                return 0;
        }
        for (int i = 2; i * i <= num; i++) {
                if (num % i == 0) {
                        return 0;
                }
        }
        return 1:
}
int linearSearchForPrime(int arr[], int size) {
        for (int i = 0; i < size; i++) {
                if (isPrime(arr[i])) {
                        return arr[i];
                }
        }
        return -1;
}
int main() {
```

```
    int size;
    printf("Enter the size of the array: ");
    scanf("%d", &size);
    int arr[size];
    printf("Enter the elements of the array:\n");
    for (int i = 0; i < size; i++) {
        scanf("%d", &arr[i]);
    }
    int result = linearSearchForPrime(arr, size);
    if (result != -1) {
        printf("First occurrence of a prime number: %d\n", result);
    }
    else {
        printf("No prime number found in the array.\n");
    }
}
```

Output:

Enter the size of the array: 6

Enter the elements of the array: 4 6 8 9 11 15

First occurrence of a prime number: 11

Program Name:

10. Write a program to take list of N Numbers ,separate even and odd numbers and put them in two appropriate file (even.txt and odd.txt).

Source Code:

```
#include <stdio.h>
int main() {
    FILE *evenFile, *oddFile;
    int N;
    printf("Enter the number of elements: ");
    scanf("%d", &N);
    int num;
    evenFile = fopen("even.txt", "w");
    oddFile = fopen("odd.txt", "w");
    if (evenFile == NULL || oddFile == NULL) {
        printf("Error opening files.\n");
        return 1;
    }
    printf("Enter the elements:\n");
    for (int i = 0; i < N; i++) {
        scanf("%d", &num);
        if (num % 2 == 0) {
            fprintf(evenFile, "%d\n", num);
        }
        else {
        fprintf(oddFile, "%d\n", num);
```

```
        }
        fclose(evenFile);
        fclose(oddFile);
        printf("Numbers separated and stored in even.txt and odd.txt.\n");
        return 0;
}
```

Output:

4

6

8

9

11

15