KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

PROGRAMMING FOR PROBLEM SOLVING LABORATOR

Subject Code : KG25ACS107

Regulation : KGR25

Academic Year : 2025-2026

I B .Tech I Sem

COMPUTER SCIENCE AND ENGINEERING

KG REDDY COLLEGE OF ENGINEERING AND TECHNOLOGY

Affiliated o JNTUH, Chilkur,(V), Moinabad(M) R. R Dist, TS-501504

KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

VISION AND MISSION OF THE INSTITUTION

VISION:

To become an institution which is internationally recognized for its holistic approach to engineering, innovative teaching and learning culture, research and entrepreneurial ecosystem, and sustainable social impact in the community.

MISSION:

• To offer undergraduate and post-graduate programs which are supported through industry relevant curriculum and innovative teaching and learning processes that would help students succeed in their professional careers.
• To provide faculty and students with an ecosystem that fosters innovation, research, entrepreneurship, and international exposure through strategic partnerships with governmentorganizations and collaboration with industries.
• To provide holistic learning environment to students which will contribute to their personal and professional growth and enable them to become leaders in their respective fields.
• To contribute to the development of the region by using our technological expertise to work with nearby communities and support them in their social and economic development.

KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISION:

To be recognized as a department of excellence by stimulating a learning environment in which students and faculty will thrive and grow to achieve their professional, institutional and societal goals.

MISSION:

- To provide high quality technical education to students that will enable life-long learning and build expertise in advanced technologies in Computer Science and Engineering.
- To promote research and development by providing opportunities to solve complex engineering problems in collaboration with industry and government agencies.
- To encourage professional development of students that will inculcate ethical values and leadership skills through entrepreneurship while working with the community to address societalissues.

PROGRAM EDUCATIONAL OBJECTIVES

PEO 1: Graduates will provide solutions to difficult and challenging issues in their profession by applying computer science and engineering theory and principles.

PEO 2: Graduates have successful careers in computer science and engineering fields or will be able to successfully pursue advanced degrees.

PEO 3: Graduates will communicate effectively, work collaboratively and exhibit high levels of professionalism, moral and ethical responsibility.

PEO 4: Graduates will develop the ability to understand and analyze engineering issues in a broader perspective with ethical responsibility towards sustainable development.

KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PROGRAM OUTCOMES

PO I: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO II: Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
PO III: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO IV: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO V: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
PO VI: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO VII: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of and need for sustainable development.
PO VIII: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO IX: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO X: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO XI: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
PO XII: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PROGRAM SPECIFIC OUTCOMES

PSO1: The Computer Science and Engineering graduates are able to analyze, design, develop, test and apply management principles, mathematical foundations in the development of intelligentsystems with computational solutions, make them to expert in designing the secure application and hardware prototype
PSO2: The graduating student will be analyze the contemporary research issues in different areasof computer science & engineering and explore research gaps, analyze and carry out research in the specialized/emerging areas.
PSO3: Develop their skills to solve problems in the broad area of programming concepts and appraise environmental and social issues with ethics and manage different projects in multi- disciplinary field to conducive in cultivating skills for successful career, entrepreneurship and higher studies.

KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Course Code: KG25ACS107 L T P C

B. Tech. I Year I – Semester 0 0 2 1

Prerequisites: A course on "Programming for Problem Solving", A course on "Computer Organization and Architecture".

Course Objectives: The objectives of this course for the student are to:

1. To work with an IDE to create, edit, compile, run and debug programs.
2. To analyze the various steps in program development
3. To develop programs to solve basic problems by understanding basicconcepts in C like operators, control statements etc.
4. To develop modular, reusable and readable C Programs using the concepts like functions, arrays etc.
5. To Write programs using the Dynamic Memory Allocation concept.
6. To create, read from and write to text and binary files

Course Outcomes: After completion of this course, the students will be able to

CO1: formulate the algorithms for simple problems

CO2: translate given algorithms to a working and correct program

CO3: correct syntax errors as reported by the compilers

CO4: identify and correct logical errors encountered during execution

CO5: represent and manipulate data with arrays, strings and structures

CO6: use pointers of different types

CO7: create, read and write to and from simple text and binary files

CO8: modularize the code with functions so that they can be reused

List of Experiments:

a. Write a simple program that prints the results of all the operators available in C (including pre/ post increment, bitwise and/or/not, etc.). Read required operand 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 from standard input.

Simple numeric problems:

a. Write a program for finding the max and min from the three numbers.

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

c. Write a 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 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

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

Expression Evaluation:

a. 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)

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

c. 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.

d. Write a C program to generate all the prime numbers between 1 and n, where n is a value supplied by the user.

e. Write a C program to find the roots of a Quadratic equation.

Arrays, Pointers and Functions:

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

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

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

following: Addition of Two Matrices

Multiplication of Two Matrices

d. Transpose of a matrix with memory dynamically allocated for the new matrix as row and column counts may not be the same.

e. 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

f. Write a program for reading elements using a pointer into an array and display the values using the array.

g. Write a program for display values reverse order from an array using a pointer.

h. Write a program through a pointer variable to sum of n elements from an 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 a C program to merge two files into 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. Write a C program to convert a Roman numeral ranging from I to L to its decimal equivalent.

b. Write a C program that uses functions to perform the following operations: To insert a sub-string into a given main string from a given position.

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

c. 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.)

d. Write a C program that displays the position of a character ch in the string S or – 1 if S

doesn‘t contain ch.

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

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

a. Write a simple program that prints the results of all the operators available in C (including pre/ post increment, bitwise and/or/not, etc.). Read required operand values from standard input.

Aim: Program that prints the results of all the operators available in C.

Program:

```
#include<stdio.h>

void main( )

{

int a,b;

printf("enter the values of a and b : ");

scanf("%d%d",&a,&b);

printf("the arithmetic operators result is %d %d %d %d \n", a+b,a-b,a*b,a/b);

printf("the relational operators result is %d %d %d %d\n", a>b,a<b,a>=b,a<=b);

printf("the logical operators result is %d %d %d %d\n", a&&b,a||b,!(a==b));

printf("the increment operator result is %d %d %d %d\n",a++,++a,b++,++b);

printf("the decrement operator result is %d %d %d %d\n",a--,--a,b--,--b);

printf("the bitwise AND operator result is %d\n",a&b);

printf("the bitwise OR operator result is %d\n",a|b);

printf("the bitwise NOT operator result is %d\n",a^b);

    }
```

Output:

```
enter the values of a and b : 10 2
the arithmetic operators result is 12 8 20 5
the relational operators result is 1 0 1 0
the logical operators result is 1 1 1
the increment operator result is 10 12 2 4
the decrement operator result is 12 10 4 2
the bitwise AND operator result is 2
the bitwise OR operator result is 10
the bitwise NOT operator result is 8
```

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

Aim: program that converts one given data type to another using auto conversion.

Program:

```
#include <stdio.h>

int main() {
    int num;
    float f;

    // Take integer input
    printf("Enter an integer: ");
    scanf("%d", &num);

    // Implicit conversion (int to float)
    f = num;

    // Explicit casting (float to int)
    int casted = (int)f;
    printf("Original integer: %d\n", num);
    printf("After implicit conversion to float: %.2f\n", f);
    printf("After explicit casting back to int: %d\n", casted);

    return 0;
}
```

Output:

```
Enter an integer: 10
Original integer: 10
After implicit conversion to float: 10.00
After explicit casting back to int: 10
```

Simple numeric problems:

a.Write a program for find the max and min from the three numbers

Aim: program for find the max and min from the three numbers.

Program:

```
#include<stdio.h>

void main( )
{
```

```
int a,b,c;

printf("Enter 3 numbers : ");
scanf("%d%d%d",&a,&b,&c);
if(a>b && a>c)
printf("Maximum number is a = %d",a);
else if(b>a && b>c)
printf("Maximum number is b = %d",b);
else
printf("Maximum number is c = %d",c);
printf("\n");
if(a<b && a<c)
printf("Minimum number is a = %d",a);
else if(b<a && b<c)
printf("Minimum number is b = %d",b);
else
printf("Minimum number is c = %d",c);

}
```

Output:

```
Enter 3 numbers : 5 20 10
Maximum number is b = 20
Minimum number is a = 5
```

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

Aim: program for the simple, compound interest.

Program:

```
#include<stdio.h>
#include<math.h>
void main()
{
int p,t;
float r,si, amount, ci;
printf("Please enter principle,time and rate of interest");
scanf("%d%d%f",&p,&t,&r);
si=p*t*r/100;
printf("\nSimple interest = %.3f",si);
amount=p*pow((1 +r/100),t);
ci=amount-p;
printf("\nCompound interest = %.3f",ci);
}
```

Output:

```
Please enter principle,time and rate of interest : 20000 5 10
```

Simple interest = 10000.000
Compound interest = 12210.203

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

Aim: program that declares class awarded for a given percentage of marks

Program:

```
#include<stdio.h>
void main()
{
int phy, chem, bio, math, comp;
float per;

printf("Enter five subjects marks: ");
scanf("%d%d%d%d%d", &phy, &chem, &bio, &math, &comp);
per = (phy + chem + bio + math + comp) / 5.0;
printf("Percentage = %.2f\n", per);
if(per >= 70)
{
printf("Distinction");
}
else if(per <70 && per>=60)
{
printf("First Class");
}
else if(per <60 && per>=40)
{
 printf("Second class");
}
else
{
printf("failed");
}
}
```

Output:

Enter five subjects marks: 85 89 89 90 95
Percentage = 89.60
Distinction

d .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
Aim: program that prints a multiplication table for a given number and the number of rows in the table.

Program:

```
 #include <stdio.h>
void main()
{
int n, i, range;
printf("Enter an integer: ");
scanf("%d",&n);
printf("Enter the range: ");
scanf("%d", &range);
for(i=1; i<=range; i++)
{
printf("%d * %d = %d \n",n,i, n*i);
}
}
```

Output:

```
Enter an integer: 6
Enter the range: 5
6* 1 = 6
6* 2 = 12
6* 3 = 18
 6* 4 = 24
  6* 5 = 30
```

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

Aim: Shows the binary equivalent of a given positive number between 0 to 255

Program:

```
#include<stdio.h>
#include<math.h>
void main()
{
int num, rem=0, inum, p=0; long int bin=0;
printf("Enter the number between 0 and 255 : ");
scanf("%d",&num);
inum=num;
while(inum>0)
{
rem=inum%2;
bin+=rem*pow(10,p);
inum/=2;
p++;
}
printf("%d binary equivalent number is %ld",num,bin);
}
```

Output:
Enter the number between 0 and 255 : 2
2 binary equivalent number is 10

Expression Evaluation:

a. 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)

Aim: program which takes two integer operands and one operator from the user, performs the operation and then prints the result

Program:

```
#include<stdio.h>
void main()
{
int a,b;
char ch;
printf(" '+' for addition\n");
printf(" '-' for subtraction\n");
printf(" '*' for multiplication\n");
printf(" '/' for division\n");
printf(" '%%' for modulus\n");
printf("Enter your choice : ");
scanf("%c",&ch);
printf("Enter two integers ");
scanf("%d%d",&a,&b);
switch(ch)
{
case '+':
printf("The addition of %d and %d is %d",a,b,a+b);
break;
case '-':
printf("The subtraction of %d and %d is %d",a,b,a-b);
break;
case '*':
printf("The multiplication of %d and %d is %d",a,b,a*b);
break;
case '/':
printf("The division of %d and %d is %d",a,b,a/b);
break;
case '%':
printf("The modulus of %d and %d is %d",a,b,a%b);
break;
default:
printf("Entered wrong operator");
}
}
```

Output:

```
'+' for addition
 '-' for subtraction
 '*' for multiplication
 '/' for division
 '%' for modulus
 Enter your choice : %
 Enter two integers 125 20
 The modulus of 125 and 20 is 5
```

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

Aim : Program to display the sum of individual digits of a positive integer and palindrome number.

Program:

```
#include <stdio.h>

void main()
{
   int num, rem, tnum, pal = 0, sum = 0;
   printf("Enter number: ");
   scanf("%d", &num);
   tnum = num;

   while (tnum > 0) {
      rem = tnum % 10;
      sum =sum+rem;
      pal = pal * 10 + rem;
      tnum =tnum/10;
   }

   printf("The sum of individual digits of %d is %d\n", num, sum);

   if (pal == num)
      printf("%d is a palindrome number\n", num);
   else
      printf("%d is not a palindrome number\n", num);
}
```

Output:

```
Enter number: 12321
The sum of individual digits of 12321 is 9
```

12321 is a palindrome number

c .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.

Aim : Program to display the first n terms of the Fibonacci sequence

Program :

```
#include <stdio.h>

int main() {
    int n, i;
    long long int t1 = 0, t2 = 1, nextTerm;

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

    if (n <= 0) {
        printf("Please enter a positive number.\n");
        return 1;
    }

    printf("Fibonacci Sequence:\n");

    for (i = 1; i <= n; i++) {
        printf("%lld ", t1);
        nextTerm = t1 + t2;
        t1 = t2;
        t2 = nextTerm;
    }

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

Output:
Enter the number of terms: 5
Fibonacci Sequence:
0 1 1 2 3

d.write a c program to generate all the prime numbers between 1 and n, where n is a value supplied by the user

Aim: program to generate all the prime numbers between 1and n.

Program:

```
#include <stdio.h>

    int main()
    {
        int n, i, j, count;

       printf("Enter n value: ");
       scanf("%d", &n);

       printf("Prime numbers between 1 and %d are:\n", n);

        for (i = 2; i <= n; i++) {
         count = 0; // Reset count for each number
                              // Count the number of divisors of i
         for (j = 1; j <= i; j++) {
         if (i % j == 0) {
            count++;
         }
      }

      // A prime number has exactly two divisors: 1 and itself
      if (count == 2) {
         printf("%d ", i);
      }
   }

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

Output :

Enter n value: 10
Prime numbers between 1 and 10 are:
1 3 5 7

e.Write a C program to find the roots of a Quadratic equation.

Aim: Program to find the roots of a Quadratic equation. **Program**:

```
#include<stdio.h>

#include<math.h>

void main()
{
float a,b,c,root1,root2;

printf("\n Enter values of a,b,c for finding roots of a quadratic eq:\n");
```

```
scanf("%f%f%f",&a,&b,&c);
/*checking condition*/
if(b*b>4*a*c)
{
root1=(-b+sqrt(b*b-4*a*c))/(2*a);
root2=(-b-sqrt(b*b-4*a*c))/(2*a);
printf("\nRoots are real\n");
printf("\n root1=%f\n root2=%f",root1,root2);
}
else if(b*b==4*a*c)
{
printf("The roots are real and equal\n");
printf("\n root1=%f\n root2=%f",root1,root2);
}
else
printf("\n Imaginary Roots.");
}
```

Output :

```
Enter values of a,b,c for finding roots of a quadratic eq:
1 34 5
Roots are real
 root1=-0.147700
 root2=-33.852299
```

Arrays , Pointers and Functions :

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

Aim: program to find the minimum, maximum and average in an array of integers.

Program:

```
#include<stdio.h>


int main()
{


int i,n,num[20],min=0,max=0;
float sum=0.0,avg=0.0;
```

```
printf("Enter how many number of elements you want to store : ");
scanf("%d",&n);

for(i=0;i<n;i++)
{
printf("Enter a[%d] value : ",i);
scanf("%d",&num[i]);
}
min = max = num[0];
for(i=0;i<n;i++)
{
sum+=num[i];
if(num[i]<min) min =num[i];

if(num[i]>max) max = num[i];
}
avg = sum/i;

printf("The minimum element of an array is %d\n",min);
printf("The Maximum element of an array is %d\n",max);
printf("The average of array elements is %f\n",avg);
return 0;
}
```

Output :

```
Enter how many number of elements you want to store : 12
Enter a[0] value : 1
Enter a[1] value : 4
Enter a[2] value : 5
Enter a[3] value : 7
Enter a[4] value : 6
Enter a[5] value : 20
```

Enter a[6] value : 10

Enter a[7] value : 8

Enter a[8] value : 6

Enter a[9] value : 9

Enter a[10] value : 3

Enter a[11] value : 45

The minimum element of an array is 1

The Maximum element of an array is 45

The average of array elements is 10.333333

b. Write a function to compute mean, variance, standard deviation , sorting of n elements in a single dimension array.

Aim : function to compute mean, variance, standard deviation and sorting of n elements in a single dimension array.

Program:

```
#include <stdio.h>
#include <math.h>  // for sqrt()

void computeAll(int arr[], int n) {

   int i, j, temp;
   float sum = 0, mean, variance = 0, stdDev;

   // Calculate Mean
   for (i = 0; i < n; i++) {
      sum += arr[i];
   }
   mean = sum / n;

   // Calculate Variance
```

```
for (i = 0; i < n; i++) {
    variance += (arr[i] - mean) * (arr[i] - mean);
  }
  variance = variance / n;   // for population variance

  // Calculate Standard Deviation
  stdDev = sqrt(variance);

  // Sort the array (Bubble Sort)
  for (i = 0; i < n - 1; i++) {
    for (j = 0; j < n - i - 1; j++) {
      if (arr[j] > arr[j + 1]) {
        temp = arr[j];
        arr[j] = arr[j + 1];
        arr[j + 1] = temp;
      }
    }
  }

  // Display results
  printf("\n--- Results ---\n");
  printf("Mean = %.2f\n", mean);
  printf("Variance = %.2f\n", variance);
  printf("Standard Deviation = %.2f\n", stdDev);


  printf("\nSorted Array: ");
  for (i = 0; i < n; i++) {
    printf("%d ", arr[i]);
  }
```

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

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

   int arr[n];

   printf("Enter %d elements:\n", n);
   for (i = 0; i < n; i++) {
      scanf("%d", &arr[i]);
   }

   computeAll(arr, n);

   return 0;
}
```

Output:

```
Enter number of elements: 5
Enter 5 elements:
5 2 8 4 6

--- Results ---
Mean = 5.00
Variance = 4.00
Standard Deviation = 2.00
Sorted Array: 2 4 5 6 8
```

c. write a c program that uses functions to perform the following

a. addition of two matrices

b. multiplication of two matrices

Aim: program that uses functions to perform addition and multiplication of two matrices.

Program :

```
#include <stdio.h>

void add(int A[10][10], int B[10][10], int C[10][10], int r, int c) {
   for (int i=0;i<r;i++)
      for (int j=0;j<c;j++)
         C[i][j] = A[i][j] + B[i][j];
}

void multiply(int A[10][10], int B[10][10], int C[10][10], int r1, int c1, int c2) {
   for (int i=0;i<r1;i++)
      for (int j=0;j<c2;j++) {
         C[i][j]=0;
         for (int k=0;k<c1;k++)
            C[i][j]+=A[i][k]*B[k][j];
      }
}

int main() {
   int A[10][10], B[10][10], C[10][10];
   int r1,c1,r2,c2,i,j;

   printf("Enter rows & cols of 1st matrix: ");
   scanf("%d%d",&r1,&c1);
   printf("Enter rows & cols of 2nd matrix: ");
   scanf("%d%d",&r2,&c2);
```

```
printf("Enter elements of A:\n");
for(i=0;i<r1;i++)
   for(j=0;j<c1;j++)
      scanf("%d",&A[i][j]);

printf("Enter elements of B:\n");
for(i=0;i<r2;i++)
   for(j=0;j<c2;j++)
      scanf("%d",&B[i][j]);

// Addition
if(r1==r2 && c1==c2) {
   add(A,B,C,r1,c1);
   printf("\nAddition:\n");
   for(i=0;i<r1;i++){
      for(j=0;j<c1;j++) printf("%d ",C[i][j]);
      printf("\n");
   }
} else printf("\nAddition not possible!\n");

// Multiplication
if(c1==r2) {
   multiply(A,B,C,r1,c1,c2);
   printf("\nMultiplication:\n");
   for(i=0;i<r1;i++){
      for(j=0;j<c2;j++) printf("%d ",C[i][j]);
      printf("\n");
   }
} else printf("\nMultiplication not possible!\n");
```

```
 return 0;
}
```

Output:

```
Enter rows & cols of 1st matrix: 2 2
Enter rows & cols of 2nd matrix: 2 2
Enter elements of A:
1 2
3 4

Enter elements of B:
5 6
7 8


Addition:
6 8
10 12

Multiplication:
19  2
43  0
```

d. Transpose of a matrix with memory dynamically allocated for the new matrix as row and column counts may not be the same .

Aim: Transpose of a matrix with memory dynamically allocated for the new matrix as row and column counts may not be same. **Program**:

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

```
#include<stdlib.h>
int main() {
    int **a, **b;
    int r, c, i, j;

    printf("Enter number of rows and columns: ");
    scanf("%d %d", &r, &c);

    // Allocate memory for matrix a (r x c)
    a = (int **)malloc(r * sizeof(int *));
    for (i = 0; i < r; i++)
        a[i] = (int *)malloc(c * sizeof(int));

    // Read elements of matrix a
    for (i = 0; i < r; i++) {
        for (j = 0; j < c; j++) {
            printf("Enter a[%d][%d]: ", i, j);
            scanf("%d", &a[i][j]);
        }
    }

    // Allocate memory for transpose matrix b (c x r)
    b = (int **)malloc(c * sizeof(int *));
    for (i = 0; i < c; i++)
        b[i] = (int *)malloc(r * sizeof(int));
    // Compute transpose
    for (i = 0; i < r; i++)
        for (j = 0; j < c; j++)
            b[j][i] = a[i][j];
```

```
// Display original matrix
   printf("\nOriginal Matrix:\n");
   for (i = 0; i < r; i++) {
      for (j = 0; j < c; j++)
         printf("%d\t", a[i][j]);
      printf("\n");
   }

   // Display transpose matrix
   printf("\nTranspose Matrix:\n");
   for (i = 0; i < c; i++) {
      for (j = 0; j < r; j++)
         printf("%d\t", b[i][j]);
      printf("\n");
   }

   return 0;
}
```

Output:

```
Enter number of rows and columns: 3 2
Enter a[0][0]: 2
Enter a[0][1]: 5
Enter a[1][0]: 8
Enter a[1][1]: 3
Enter a[2][0]: 4
```

```
Enter a[2][1]:9
Original Matrix:
2   5
8   3
4   9

Transpose Matrix:
2   8   4
5   3   9
```

e. 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

A. I.Aim: Recursive function to find the factorial of a given integer**.**

Program:

```
#include <stdio.h>

// Recursive function to find factorial long long factorial(int n) {
    if (n == 0 || n == 1)
        return 1;           // Base case else
        return n * factorial(n - 1);  // Recursive call
}

int main() {
int num;
    printf("Enter a number: ");
    scanf("%d", &num);

    printf("Factorial of %d = %lld\n", num, factorial(num));
    return 0;

}
```

Output :

Enter a number: 5
Factorial of 5 =120

II.**Aim**:Non-Recursive function to find the factorial of a given integer.

Program:

```
#include <stdio.h>

int main() { int n, i;
    long long fact = 1;

    printf("Enter a number: ");
     scanf("%d", &n);

    for (i = 1; i <= n; i++) {
        fact = fact * i;   // Multiply each number from 1 to n
    }

    printf("Factorial of %d = %lld\n", n, fact);
     return 0;
}
```

Output:

Enter a number: 5
Factorial of 5 =120

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

I.Aim: Recursive GCD – Euclid's Algorithm Program:

```
#include <stdio.h>

// Recursive function to find GCD
 int gcd(int a, int b) {
    if (b == 0) return a;
    return gcd(b, a % b);
}
int main() { int x, y;
    printf("Enter two integers: ");

    scanf("%d %d", &x, &y);
```

```
printf("GCD = %d\n", gcd(x, y));
 return 0;

}
```

Output:

Enter two integers: 8 6
GCD = 2

II.Aim: Non-Recursive GCD

Program:

```
#include <stdio.h>

int main() {
int a, b, temp;

printf("Enter two numbers: ");
 scanf("%d %d", &a, &b);

                    while (b != 0) {
                        temp = b;
                    b = a % b;
                     a = temp;
}

printf("GCD = %d\n", a);
return 0;
}
```

Output:

Enter two integers: 8 6
GCD = 2

C. I. Recursive Power Function

Program:

```
#include <stdio.h>

// Recursive function to find x^n
long long power(int x, int n) {
if (n == 0)
 return 1;
```

```
    return x * power(x, n - 1);   // Recursive step
    }

int main() { int x, n;

    printf("Enter value of x: ");
    scanf("%d", &x);

    printf("Enter value of n: ");
    scanf("%d", &n);

    printf("%d^%d = %lld\n", x, n, power(x, n));
    return 0;
    }
```

OutPut:

```
Enter value of x: 2
Enter value of n: 3
2^3 = 8
```

II. Non-Recursive Power Function Program:

```
#include <stdio.h>
int main() {
 int x, n, i;
long long result = 1;

printf("Enter value of x: ");
scanf("%d", &x);
printf("Enter value of n: "); scanf("%d", &n);

for (i = 1; i <= n; i++) { result = result * x;
}

printf("%d^%d = %lld\n", x, n, result); return 0;
}
```

OutPut:

```
Enter value of x: 2
Enter value of n: 3
2^3 = 8
```

f. Write a program for reading elements using a pointer into an array and display the values using the array.

Aim: program for reading elements using a pointer into an array and display the values using the array.

Program:

```
#include <stdio.h>

 int main() {

int n;


printf("Enter number of elements: ");


 scanf("%d", &n);



 int arr[n];   // Declare an array of size n

 int *ptr = arr;  // Pointer pointing to the first element of the array
// Reading elements using pointer

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

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

 {

scanf("%d", ptr + i);   // Pointer arithmetic: ptr + i points to arr[i]

}

// Displaying elements using array notation

 printf("The elements in the array are:\n");

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

    printf("%d ", arr[i]);  // Using array indexing to display

}

printf("\n");
```

```
    return 0;
}
```

Output :

Enter number of elements: 4

Enter 4 elements:

5

4

3

20

The elements in the array are:

5 4 3 20

g. Write a program for display values reverse order from an array using a pointer.

Aim: display values reverse order from an array using a pointer.

Program:

```
#include <stdio.h>

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

    int arr[n]; // Declare an array
    int *ptr = arr;          // Pointer pointing to the first element

    // Reading elements using array notation
    printf("Enter %d elements:\n", n);
    for (int i = 0; i < n; i++)
```

```
    {
     scanf("%d", &arr[i]);
    }
    // Displaying elements in reverse using pointer

    printf("Elements in reverse order:\n");
    for (int i = n - 1; i >= 0; i--) {
        printf("%d ", *(ptr + i));  // Pointer arithmetic: access arr[i] via ptr
    }
    printf("\n");


    return 0;
}
```

Output:

Enter number of elements: 4

Enter 4 elements:

4

5

8

9

Elements in reverse order:

9 8 5 4

h. Write a program through a pointer variable to sum of n elements from an array.

Aim: Sum of n Elements Using Pointer

Program :

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

```
int main() {
 int n;
printf("Enter number of elements: ");
 scanf("%d", &n);
int arr[n]; // Declare an array
int *ptr = arr;           // Pointer pointing to the first element

// Reading elements into the array
 printf("Enter %d elements:\n", n);
 for (int i = 0; i < n; i++) {
scanf("%d", ptr + i);  // Reading using pointer arithmetic
}

// Calculating sum using pointer
 int sum = 0;
for (int i = 0; i < n; i++) {
sum += *(ptr + i);  // Access each element using pointer
}
printf("Sum of the elements = %d\n", sum);

return 0;
}
```

Output:

Enter number of elements: 4

Enter 4 elements:

5

7

8

6

Sum of the elements = 26

Files:

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

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

Program:

```
#include <stdio.h>
 #include <stdlib.h>
int main(int argc, char *argv[])
{
FILE *fs;
 int ch;
/* Check number of arguments */
if (argc != 2)
{
printf("Usage: %s <filename>\n", argv[0]);
 return 1;
}
/* Open file in read mode */

fs = fopen(argv[1], "r");
if (fs == NULL)
{
printf("Source file cannot be opened.\n");
return 1;
}
/* Read and display file contents */
 while ((ch = fgetc(fs)) != EOF)
{
```

```
putchar(ch);
}
/* Close file */
fclose(fs);
return 0;
}
```

Output:

C:\users\lenovo\desktop\sravani\sample.txt: welcome to c programming

C:\users\lenovo\desktop\sravani\display.exe: welcome to c programming

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

Aim: C program which copies one file to another, replacing all lowercase characters with their uppercase equivalents

Program :

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
char ch;
FILE *fp1, *fp2;
/* Open source file */
fp1 = fopen("cbi.txt", "r");
if (fp1 == NULL)
{
printf("Source file cannot be opened.\n");
return 1;
}
/* Open destination file */
```

```
fp2 = fopen("lmn.txt", "w");
if (fp2 == NULL)
{
printf("Destination file cannot be opened.\n");
 fclose(fp1);
return 1;
}
/* Read from source and write to destination */
while ((ch = fgetc(fp1)) != EOF)
{
if (ch >= 'a' && ch <= 'z')
{
ch = ch - 32;  // convert to uppercase
}
fputc(ch, fp2);
}
fclose(fp1);
fclose(fp2);
printf("File copied successfully with uppercase conversion.\n");
return 0;
}
```

OutPut:

C:\users\lenovo\desktop\sravani\cbi.txt: Fish in the water

C:\users\lenovo\desktop\sravani\copy_upper.exe: File copied successfully with uppercase conversion.

C:\users\lenovo\desktop\sravani\imn.txt: FISH IN THE WATER

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.

Aim: C program to count the number of times a character occurs in a text file.

Program:

```
#include <stdio.h>
 #include <stdlib.h>
int main(int argc, char *argv[])
{
    int ch;
    int count = 0;
    FILE *fp1;

    if (argc != 3)
    {
        printf("Usage: %s <filename> <character>\n", argv[0]);
        return 1;
    }

    fp1 = fopen(argv[1], "r");

    if (fp1 == NULL)
    {
        printf("File cannot be opened.\n");
        return 1;
    }

    char search_char = argv[2][0];  // first character of 2nd argument


    while ((ch = fgetc(fp1)) != EOF)
    {
        if ((char)ch == search_char)
        {
            count++;
        }
    }

    fclose(fp1);


    printf("The character '%c' occurred %d times in the file %s.\n", search_char, count, argv[1]);

    return 0;
    }
```

OutPut:

C:\users\lenovo\desktop\sravani\data.txt: fish in the water

C:\users\lenovo\desktop\sravani\count.exe: The character 'i' occurred 2 time in the file data.txt

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

Aim: C program to merge two files into a third file.

Program:

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[])
{
FILE *f1, *f2, *f3;
int ch;
if (argc != 4)
{
   printf("Usage: %s <source1> <source2> <target>\n", argv[0]);
    return 1;
}
/* Open first source file */
f1 = fopen(argv[1], "r");
if (f1 == NULL)
{
   printf("Source file %s cannot be opened.\n", argv[1]);
   return 1;
}
/* Open second source file */
f2 = fopen(argv[2], "r");
if (f2 == NULL)
{
printf("Source file %s cannot be opened.\n", argv[2]);
fclose(f1);
```

```
    return 1;
    }

/* Open target file */
f3 = fopen(argv[3], "w");
if (f3 == NULL)
{

printf("Target file %s cannot be opened.\n", argv[3]);
 fclose(f1);
fclose(f2);
return 1;
}
/* Copy contents of first source file */
 while ((ch = fgetc(f1)) != EOF)
{
fputc(ch, f3);
}

/* Copy contents of second source file */
while ((ch = fgetc(f2)) != EOF)
{
fputc(ch, f3);
}
/* Close all files */
 fclose(f1);
 fclose(f2);
 fclose(f3);

printf("Files merged successfully into %s\n", argv[3]);

return 0;
```

```
}
```

Output:

C:\users\lenovo\desktop\sravani\data1.txt: Happy

C:\users\lenovo\desktop\sravani\data2.txt: Birthday

In parameters type:data1.txt data2.txt destination.txt

C:\users\lenovo\desktop\sravani\merge.exe: Files merged successfully into destination.txt

C:\users\lenovo\desktop\sravani\destination.txt: Happy Birthday

Strings:

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

Aim: C program to convert a Roman numeral ranging from I to L to its decimal equivalent

Program:

```
#include <stdio.h>
 #include <string.h>
#include <stdlib.h>
int romanToDecimal(char *rom)
{
 int len = strlen(rom);
int *a = (int *)malloc(len * sizeof(int)); // Dynamically allocate memory for the array
int k = 0;
for (int i = 0; i < len; i++) {
if (rom[i] == 'I')
 a[i] = 1;
else if (rom[i] == 'V') a[i] = 5;
else if (rom[i] == 'X') a[i] = 10;
else if (rom[i] == 'L') a[i] = 50;
else if (rom[i] == 'C') a[i] = 100;
else if (rom[i] == 'D') a[i] = 500;
else if (rom[i] == 'M') a[i] = 1000; else {
```

```
printf("Invalid Roman Numeral\n");
free(a); // Free the allocated memory before exiting return -1;
}
}

k = a[len - 1];
for (int i = len - 1; i > 0; i--) {
if (a[i] > a[i - 1])
k = k - a[i - 1];
else if (a[i] <= a[i - 1]) k = k + a[i - 1];
}
free(a); // Free the allocated memory
return k;
}
int main() {
char rom[100]; // Array to store the Roman numeral input
printf("Enter the Roman Numeral: ");
scanf("%s", rom);
int decimal = romanToDecimal(rom);
if (decimal != -1) {
printf("\nIts Decimal Equivalent is: %d\n", decimal);
}
return 0;
}
```

Output:

Enter the Roman Numeral: XI

Its Decimal Equivalent is: 11

b. Write a C program that uses functions to perform the following operations:

a. To insert a sub-string into a given main string from a given position.

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

a. Aim: To insert a sub-string into a given main string from a given position.

Program:

```
#include <stdio.h>
#include <string.h>
void main() {
char a[40]; // Increased size to avoid overflow during insertion
char b[20];
char c[40];
int p = 0, r = 0, i = 0;

int t = 0, x, g, s, n, o;
printf("Enter First String: ");
fgets(a, sizeof(a), stdin); // safer alternative to gets()

// Remove newline character if fgets() reads it a[strcspn(a, "\n")] = '\0';

printf("Enter Second String: ");
fgets(b, sizeof(b), stdin); // safer alternative to gets()



// Remove newline character if fgets() reads it
 b[strcspn(b, "\n")] = '\0';

printf("Enter the position where the item has to be inserted: ");
scanf("%d", &p);

r = strlen(a); // Length of the first string
n = strlen(b); // Length of the second string
if (p < 0 || p > r) {
printf("Invalid position.\n");
return;
}
// Copying the first string into the result string
for (i = 0; i < r; i++) {
```

```
c[i] = a[i];
}
// Inserting second string at the specified position
for (i = r - 1; i >= p; i--) {
a[i + n] = a[i]; // Shift characters in a to make room
}
// Copy second string into the new position
for (i = 0; i < n; i++) {
a[p + i] = b[i];
}

// Print the result
printf("Resulting String: %s\n", a);
}
```

Output:

```
Enter First String: problem
Enter Second String: solving
Enter the position where the item has to be inserted: 4
Resulting String: probsolvinglem
```

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

Aim: To delete n Characters from a given position in a given string.

Program:

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

/* Function prototype */
void delchar(char *x, int n, int pos);


int main()
{
char string[50]; int n, pos;
```

```
printf("Enter the string: ");
fgets(string, sizeof(string), stdin);


/* Remove newline added by fgets */ string[strcspn(string, "\n")] = '\0';

printf("Enter the position from where to delete: ");
scanf("%d", &pos);
printf("Enter the number of characters to be deleted: ");
scanf("%d", &n);
delchar(string, n, pos);



return 0;
}
/* Function to delete n characters */
void delchar(char *x, int n, int pos)
{
int len = strlen(x);
if (pos < 1 || pos > len) {
 printf("Invalid position\n");
 return;
}
if (pos + n - 1 > len) {
printf("Deletion exceeds string length\n");
 return;
}
strcpy(&x[pos - 1], &x[pos - 1 + n]);
printf("Resulting string: %s\n", x);
}
```

Output:

```
Enter the string: CProgramming
```

Enter the position from where to delete: 1

Enter the number of characters to be deleted: 1

Resulting string: Programming

c. 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.)

Aim: determine if the given string is a palindrome or not

Program:

```
#include <stdio.h>
#include <string.h>
int main()
{
char str[50], strrev[50];
 int i, j, len;
printf("Enter a string: ");
 scanf("%s", str);


len = strlen(str);
/* First loop: set j to 0 */
 j = 0;
/* Second loop: reverse the string */
for (i = len - 1; i >= 0; i--)
{
strrev[j] = str[i]; j++;
}
strrev[j] = '\0';
if (strcmp(str, strrev) == 0)
printf("%s is a palindrome\n", str); else
printf("%s is not a palindrome\n", str);
return 0;
}
```

Output:

Enter a string: mam

mam is a palindrome

d. Write a C program that displays the position of a character ch in the string S or – 1 if S doesn‘t contain ch.

Aim: display the position of a character ch in the string S or – 1 if S doesn‘t contain ch

Program:

```
#include <stdio.h>
#include <string.h>
int main()
{
char s[30]; char t;
char *found;
/* Entering the main string */
printf("Enter the first string: ");
fgets(s, sizeof(s), stdin);

/* Remove newline from fgets */
s[strcspn(s, "\n")] = '\0';

/* Entering the character to be searched */
printf("Enter the character to be searched: ");
scanf(" %c", &t);  // space before %c ignores newline
/* Searching character t in string s */
found = strchr(s, t);
if (found != NULL)
printf("Character found at position %ld\n", found - s);
else
printf("-1\n");
return 0;
}
```

Output:

Enter the first string: program

Enter the character to be searched: m

Character found at position 6

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

Aim: C program to count the lines, words and characters in a given text.

Program:

```
#include <stdio.h>
#include <string.h>
int main()
{
char line[81];
 int i;
int characters = 0, words = 0, lines = 0;
printf("KEY IN THE TEXT.\n");
printf("GIVE ONE SPACE AFTER EACH WORD.\n");
printf("WHEN COMPLETED, PRESS ENTER ON AN EMPTY LINE.\n\n");
while (1)
{
/* Read a line */
if (fgets(line, sizeof(line), stdin) == NULL) break;
/* Stop when empty line is entered */
if (strcmp(line, "\n") == 0) break;

lines++;
/* Count characters (excluding newline) */
characters += strlen(line) - 1;

/* Count words */
words++; // first word
for (i = 0; line[i] != '\0'; i++)
{
```

```
if (line[i] == ' ' || line[i] == '\t') words++;
}
}

printf("\nNumber of lines = %d\n", lines);
 printf("Number of words = %d\n", words);
printf("Number of characters = %d\n", characters);
return 0;
}
```

OutPut:

```
KEY IN THE TEXT.
GIVE ONE SPACE AFTER EACH WORD.
WHEN COMPLETED, PRESS ENTER ON AN EMPTY LINE.
hello world
this is c programming this is c programming
Number of lines = 3
 Number of words = 11
 Number of characters = 54
```

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.

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

Program:

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_LEN 10
void l_search_nonrecursive(int l[], int num, int ele);
int main()
{
```

```
int l[MAX_LEN], num, ele, i;
printf("Enter the number of elements: ");
scanf("%d", &num);
if (num > MAX_LEN)
{
printf("Maximum allowed elements are %d\n", MAX_LEN);
return 0;
}
printf("\nEnter the elements:\n");
 for (i = 0; i < num; i++)
scanf("%d", &l[i]);
printf("\nElements present in the list are:\n");
for (i = 0; i < num; i++)
printf("%d\t", l[i]);
printf("\n\nElement you want to search: ");
 scanf("%d", &ele);
l_search_nonrecursive(l, num, ele);
return 0;
}
/* Non-Recursive Linear Search */
void l_search_nonrecursive(int l[], int num, int ele)
{
int i;
for (i = 0; i < num; i++)
{
if (l[i] == ele)
{

printf("\nThe element %d is present at index position %d\n", ele, i);
return;
}
}
```

```
printf("\nThe element %d is not present in the list\n", ele);
}
```

Output:

Enter the number of elements: 4 Enter the elements:

10 50 80 70

Elements present in the list are:

10 50 80 70

Element you want to search: 70

The element 70 is present at index position 3

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.

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

Program:

```
#include <stdio.h>
 #include <stdlib.h>
#define MAX_LEN 10
void b_search_nonrecursive(int l[], int num, int ele);
/* Main function */
 int main()
{
int l[MAX_LEN], num, ele, i;
printf("Enter the number of elements: ");
scanf("%d", &num);
if (num > MAX_LEN)
{
printf("Maximum allowed elements are %d\n", MAX_LEN);
 return 0;
}
printf("\nEnter the elements in SORTED order:\n");
```

```
 for (i = 0; i < num; i++)
scanf("%d", &l[i]);
printf("\nElements present in the list are:\n");
for (i = 0; i < num; i++)
printf("%d\t", l[i]);
printf("\n\nEnter the element you want to search: ");
 scanf("%d", &ele);
b_search_nonrecursive(l, num, ele);
return 0;
}
/* Non-Recursive Binary Search Function */
void b_search_nonrecursive(int l[], int num, int ele)
{
int low = 0, high = num - 1, mid;
while (low <= high)
{
mid = (low + high) / 2;
if (l[mid] == ele)
{
printf("\nThe element %d is present at index position %d\n", ele, mid);
 return;
}
else if (l[mid] < ele) low = mid + 1;
else
high = mid - 1;
}
printf("\nThe element %d is not present in the list\n", ele);
}
```

OutPut:

Enter the number of elements: 6

Enter the elements in SORTED order: 10 20 30 40 50 60

Elements present in the list are:

10 20 30 40 50 60

Enter the element you want to search: 50

The element 50 is present at index position 4

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

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

Program:

```
#include <stdio.h>
#define MAX 10
/* Function to swap two numbers */
 void swapList(int *m, int *n)
{
int temp = *m;
*m = *n;
*n = temp;
}
/* Function for Bubble Sort */
void bub_sort(int list[], int n)
{
int i, j;
for (i = 0; i < n - 1; i++)
{
for (j = 0; j < n - i - 1; j++)
```

```
{
if (list[j] > list[j + 1])

swapList(&list[j], &list[j + 1]);
}

}
}
/* Function to read list elements */
void readlist(int list[], int n)
{
int j;
printf("\nEnter the elements:\n");
 for (j = 0; j < n; j++)
scanf("%d", &list[j]);
}
/* Function to print list elements */
void printlist(int list[], int n)
{
int j;
for (j = 0; j < n; j++)
printf("%d\t", list[j]);
}
/* Main function */
 int main()
{
int list[MAX], num;
printf("Enter the number of elements (Maximum %d): ", MAX);
 scanf("%d", &num);
if (num > MAX || num <= 0)
{
printf("Invalid number of elements.\n");
```

```
return 0;
}
readlist(list, num);

printf("\nElements in the list before sorting are:\n");
 printlist(list, num);
bub_sort(list, num);
printf("\n\nElements in the list after sorting are:\n");
printlist(list, num);

return 0;
}
```

OutPut:

```
Enter the number of elements (Maximum 10): 5
Enter the elements:
10 5 8 4 1
Elements in the list before sorting are:
  10      5         8         4         1
Elements in the list after sorting are:
1     4        5      8     10
```

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

Aim: program that sorts the given array of integers using selection sort in descending order**.**

Program:

```
#include <stdio.h>
#define MAXSIZE 500
void selection(int elements[], int maxsize);
int main()
{
int elements[MAXSIZE];
 int maxsize, i;
printf("How many elements do you want to sort: ");
```

```
scanf("%d", &maxsize);
if (maxsize <= 0 || maxsize > MAXSIZE)
{
printf("Invalid number of elements.\n");

return 0;
}
printf("\nEnter the values one by one:\n");
for (i = 0; i < maxsize; i++)
{
printf("Enter element %d: ", i);
 scanf("%d", &elements[i]);
}



printf("\nArray before sorting:\n");
 for (i = 0; i < maxsize; i++)
printf("%d\t", elements[i]);
selection(elements, maxsize);
printf("\n\nArray after sorting:\n");
 for (i = 0; i < maxsize; i++)
printf("%d\t", elements[i]);
return 0;
}
/* Selection Sort Function */
void selection(int elements[], int maxsize)
{
int i, j, min, temp;
for (i = 0; i < maxsize - 1; i++)
{
min = i;
for (j = i + 1; j < maxsize; j++)
```

```
{
if (elements[j] < elements[min]) min = j;
}
temp = elements[i];
elements[i] = elements[min];

elements[min] = temp;
}
}
```

Output:

```
How many elements do you want to sort: 5
Enter the values one by one: Enter element 0: 7
Enter element 1: 8
Enter element 2: 6
Enter element 3: 4
Enter element 4: 43
Array before sorting:
7       8       6       4       43
Array after sorting:
4   6   7       8       43
```

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

Aim: C program that sorts the given array of integers using insertion sort in ascending order

Program:

```
#include <stdio.h>
int main()
{
int n, i, j, temp; int arr[64];
printf("Enter number of elements: ");
scanf("%d", &n);
if (n <= 0 || n > 64)
{
printf("Invalid number of elements.\n");
```

```
return 0;
}
printf("Enter %d integers:\n", n);
for (i = 0; i < n; i++)

{
scanf("%d", &arr[i]);
}
/* Insertion Sort Logic */
for (i = 1; i <= n - 1; i++)
{
j = i;
while (j > 0 && arr[j - 1] > arr[j])
{
temp = arr[j];
arr[j] = arr[j - 1]; arr[j - 1] = temp;
 j--;
}
}

printf("\nSorted list in ascending order:\n");
for (i = 0; i <= n - 1; i++)
{
printf("%d\n", arr[i]);
}
return 0;
}
```

OutPut:

Enter number of elements: 5 Enter 5 integers:

5 8 9 7 6

Sorted list in ascending order:

5

6

7

8

9

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

Aim: C program that sorts a given array of names

Program:

```
#include <stdio.h>
#include <string.h>
#define MAX_NAMES 20
#define MAX_LEN  20
int main()
{
int i, j, num;
char name[MAX_NAMES][MAX_LEN];
char t_name[MAX_NAMES][MAX_LEN];
char temp[MAX_LEN];
printf("Enter how many names you want to sort: ");
 scanf("%d", &num);
if (num <= 0 || num > MAX_NAMES)
{
printf("Invalid number of names.\n");

return 0;
}
printf("Enter %d names one by one:\n", num);
for (i = 0; i < num; i++)
{
scanf("%s", name[i]);
strcpy(t_name[i], name[i]); // store original list
}
```

```
/* Sorting names alphabetically */
 for (i = 0; i < num - 1; i++)
{
for (j = i + 1; j < num; j++)
{

if (strcmp(name[i], name[j]) > 0)
{
strcpy(temp, name[i]);
strcpy(name[i], name[j]);
 strcpy(name[j], temp);
}
}
}
printf("\nNames before sorting in alphabetical order:\n");
 for (i = 0; i < num; i++)
printf("%s\n", t_name[i]);
printf("\nNames after sorting in alphabetical order:\n");
 for (i = 0; i < num; i++)
printf("%s\n", name[i]);
return 0;
}
```

Output:

Enter how many names you want to sort: 3

Enter 3 names one by one:

KG REDDY
College of Engineering
& Technology
AN AUTONOMOUS INSTITUTION