KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

C PROGRAMMING & DATA STRUCTURES LABORATORY

Subject Code : KG25ACS107ES

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: KG25ACS107ES L T P C

B. Tech. I Year I – Semester 0 0 2 1

Prerequisites: A course on " C Programming & Data Structures".

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

1. Apply C programming constructs to develop modular and structured solutions for computational problems.
2. Analyze and implement string handling, function-based modularization, and file operations in C.
3. Design and implement basic data structures such as stacks, queues, and linked lists to solve problems efficiently.
4. Implement and compare searching and sorting techniques for optimal data handling.
5. Enhance logical thinking, debugging proficiency, and self-learning through hands-on experiments and mini-projects.

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

CO1: Develop modular and readable C Programs.

CO2: Solve problems using strings, functions.

CO3. Handle data in files.

CO4. Implement stacks, queues using arrays, linked lists.

C05. To understand and analyze various searching and sorting algorithms.

List of Experiments:

1. Write a C program to find the sum of individual digits of a positive integer.

2. 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.

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

4. Write a C program to find the roots of a quadratic equation.

5. Write a C program to find the factorial of a given integer.

6. Write a C program to find the GCD (greatest common divisor) of two given integers.

7. Write a C program to solve Towers of Hanoi problem.

8. 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).

9. Write a C program to find both the largest and smallest number in a list of integers.

10. Write a C program that uses functions to perform the following:
    i) Addition of Two Matrices
    ii) Multiplication of Two Matrices

11. Write a C program that uses functions to perform the following operations:
    i) To insert a sub-string in to a given main string from a given position.
    ii) To delete n Characters from a given position in a given string.

12. Write a C program to determine if the given string is a palindrome or not.

13. Write a C program that displays the position or index in the string S where the string T begins, or – 1 if S doesn't contain T.

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

15. Write a C program to generate Pascal's triangle.

16. Write a C program to construct a pyramid of numbers.

17. Write a C program that uses functions to perform the following operations:
    i) Reading a complex number
    ii) Writing a complex number
    iii) Addition of two complex numbers
    iv) Multiplication of two complex numbers
    (Note: represent complex number using a structure.)

18. i) Write a C program which copies one file to another.
    ii) Write a C program to reverse the first n characters in a file.
    (Note: The file name and n are specified on the command line.)

19. i) Write a C program to display the contents of a file.
    ii. 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).

20. Write a C program that uses functions to perform the following operations on singly linked list:
    i) Creation ii) Insertion iii) Deletion iv) Traversal

21. Write C programs that implement stack (its operations) using
    i) Arrays
    ii) Pointers

22. Write C programs that implement Queue (its operations) using
    i) Arrays
    ii) Pointers

23. Write a C program that implements the following sorting methods to sort a given list of integers in ascending order
    i) Bubble sort
    ii) Selection sort
    iii) Insertion sort

24. Write C programs that use both recursive and non-recursive functions to perform the following searching operations for a Key value in a given list of integers:
    i) Linear search
    ii) Binary search

1. Write a C program to find the sum of individual digits of a positive integer.

Aim: C program to find the sum of individual digits of a positive integer.

Program:

```
#include <stdio.h>

int main() {

int num, sum = 0, digit;

printf("Enter a positive integer: ");

scanf("%d", &num);

// Loop to extract digits and add them

while (num > 0) {

digit = num % 10;   // get last digit

sum += digit;       // add to sum

num = num / 10;     // remove last digit

}

printf("Sum of digits = %d\n", sum);

return 0;

}
```

Output:

```
Enter a positive integer: 12345

Sum of digits = 15
```

2. 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, first = 0, second = 1, next, i;
  // Input: number of terms
  printf("Enter the number of terms: ");
  scanf("%d", &n);
  printf("Fibonacci Sequence:\n");
  for (i = 1; i <= n; i++) {
  printf("%d ", first); // Print current term
  next = first + second; // Compute next term
  first = second;         // Update first

second = next;           // Update second
  }
  printf("\n");

return 0;
  }
```

Output:

Enter the number of terms: 7

Fibonacci Sequence:

0 1 1 2 3 5 8

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

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

Program:

```
include <stdio.h>

int main(){

int n, i, j, count;

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

scanf("%d", &n);

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

for (i = 2; i <= n; i++) {

count = 0; // reset for each number

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

if (i % j == 0)

count++;

}

if (count == 2) // only divisible by 1 and itself

printf("%d ", i);

}
```

```
return 0;

}
```

Output:

```
Enter the value of n: 20

Prime numbers between 1 and 20 are:

2 3 5 7 11 13 17 19
```

4. Write a C program to find the roots of a quadratic equation.

Aim: C program To find the roots of a Quadratic equation

Program:

```
#include <stdio.h>

#include <math.h>

int main() {
        float a, b, c, d, r1, r2;
        printf("Enter values of a, b and c: ");
        scanf("%f %f %f", &a, &b, &c);
        d = b * b - 4 * a * c; // discriminant
  if (d > 0) {
                r1 = (-b + sqrt(d)) / (2 * a);
                r2 = (-b - sqrt(d)) / (2 * a);
                printf("Roots are real and different.\n");
                printf("Root1 = %.2f\nRoot2 = %.2f\n", r1, r2);
        }
        else if (d == 0) {
                r1 = r2 = -b / (2 * a);
                printf("Roots are real and equal.\n");
                printf("Root1 = Root2 = %.2f\n", r1);
        }
        else {
                printf("Roots are imaginary.\n");
        }
        return 0;
  }
```

Output:

Enter values of a, b and c: 1 -3 2

Roots are real and different.

Root1 = 2.00

Root2 = 1.00

5. Write a C program to find the factorial of a given integer.

Aim: C program 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;
   }

printf("Factorial of %d = %lld\n", n, fact);

   return 0;
   }
```

Output:

```
Enter a number:5
   Factorial of 5 = 120
```

6. Write a C program to find the GCD (greatest common divisor) of two given integers.

Aim: C program to find the GCD (greatest common divisor) of two given integers

Program:

```
#include <stdio.h>

int main() {
   int n1, n2, i, gcd;
   printf("Enter two numbers: ");
   scanf("%d %d", &n1, &n2);
   for(i = 1; i <= n1 && i <= n2; i++) {
   if(n1 % i == 0 && n2 % i == 0)
   gcd = i;
   }
                        printf("GCD of %d and %d = %d\n", n1, n2, gcd);

   return 0;
   }
```

Output:

Enter two numbers: 5 10

GCD of 5 and 10 = 5

7. Write a C program to solve Towers of Hanoi problem.

Aim: C program to solve Towers of Hanoi problem.

Program:

```
#include <stdio.h>

void hanoi(int n, char from, char aux, char to); // function declaration

int main() {

int n;

printf("Enter number of disks: ");

scanf("%d", &n);

hanoi(n, 'A', 'B', 'C');

return 0;

}

void hanoi(int n, char from, char aux, char to) {

if (n == 1) {

printf("Move disk 1 from %c to %c\n", from, to);

return;

}

hanoi(n - 1, from, to, aux);

printf("Move disk %d from %c to %c\n", n, from, to);

hanoi(n - 1, aux, from, to);
```

```
}
```

Output:

```
Enter number of disks: 3

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
```

8. 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

9. Write a C program to find both the largest and smallest number in a list of integers.

Aim: C program to find both the largest and smallest number in a list of integers.

Program:

```
#include <stdio.h>

int main() {
   int n, i;
   int arr*100+;
   int largest, smallest;

printf("Enter how many integers: ");

scanf("%d", &n);

printf("Enter the integers:\n");

for (i = 0; i < n; i++) {
        scanf("%d", &arr*i+);
   }

// initialize

   largest = arr*0+;
        smallest = arr*0+;
   // check each number
        for (i = 1; i < n; i++) {
                if (arr*i+ > largest)
                        largest = arr*i+;
                if (arr*i+ < smallest)
                        smallest = arr*i+;
        }

printf("\nLargest number = %d\n", largest);
```

```
printf("Smallest number = %d\n", smallest);

return 0;
  }
```

Output:

```
  Enter how many integers: 5
  Enter the integers:

50

650

10

2

3

Largest number = 650

Smallest number = 2
```

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

i) Addition of Two Matrices

ii) Multiplication of Two Matrices

i) Addition of two Matrices

Aim: C program that uses functions to perform Addition of Two Matrices

Program:

```
#include <stdio.h>

int main() {
   int rows, cols, i, j;
   int matrix1[10][10], matrix2[10][10], sum[10][10];

printf("Enter the number of rows and columns of the matrices: ");
   scanf("%d %d", &rows, &cols);

// Input first matrix
   printf("Enter the elements of first matrix:\n");
   for (i = 0; i < rows; i++)
         for (j = 0; j < cols; j++)
               scanf("%d", &matrix1[i][j]);

// Input second matrix
   printf("Enter the elements of second matrix:\n");
   for (i = 0; i < rows; i++)
         for (j = 0; j < cols; j++)
               scanf("%d", &matrix2[i][j]);

// Add matrices
   printf("Sum of the two matrices:\n");
   for (i = 0; i < rows; i++) {
         for (j = 0; j < cols; j++) {
```

```
sum[i][j] = matrix1[i][j] + matrix2[i][j];
                    printf("%d\t", sum[i][j]);
            }


printf("\n");
  }

  return 0;
  }
```

Output:

```
Enter the number of rows and columns of matrix: 2 2
  Enter the elements of first matrix:
  1 2
  3 4
  Enter the elements of second matrix:
  5 6
  7 8
  Sum of entered matrices:
  6 8
  10 12
```

ii) Multiplication of Two Matrices

Aim: C program that uses functions to perform Addition of Two Matrices

Program:

```
#include <stdio.h>

int main() {
   int i, j, k, m, n, p, q;
   int a[10][10], b[10][10], c[10][10];

printf("Input rows and columns of A matrix: ");
   scanf("%d %d", &m, &n);

printf("Input rows and columns of B matrix: ");
   scanf("%d %d", &p, &q);

if (n != p) {
          printf("Matrices cannot be multiplied! Columns of A must equal rows of B.\n");

   return 0;
   }

printf("Matrices can be multiplied. Resultant matrix will be %d x %d\n", m, q);

// Input A matrix
   printf("Input A matrix:\n");
   for (i = 0; i < m; i++)
          for (j = 0; j < n; j++) {
                        printf("Enter a[%d][%d] value: ", i, j);
                        scanf("%d", &a[i][j]);
                  }// Input B matrix
   printf("Input B matrix:\n");
   for (i = 0; i < p; i++)
```

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

printf("Enter b[%d][%d] value: ", i, j);

scanf("%d", &b[i][j]);
                }

// Multiply matrices
   for (i = 0; i < m; i++) {
          for (j = 0; j < q; j++) {
                c[i][j] = 0;
                for (k = 0; k < n; k++) {
                      c[i][j] += a[i][k] * b[k][j];

}
          }
   }

// Display result
   printf("\n===== Matrix Multiplication Result =====\n");
   for (i = 0; i < m; i++) {
          for (j = 0; j < q; j++)
                printf("%d\t", c[i][j]);
          printf("\n");
   }

   return 0;
   }
```

Output:

Input rows and columns of A matrix: 3 3

Input rows and columns of B matrix: 3 3

Matrices can be multiplied. Resultant matrix will be 3 x 3

Input A matrix:

Enter a[0][0] value: 2

Enter a[0][1] value: 3

Enter a[0][2] value: 4

Enter a[1][0] value: 5

Enter a[1][1] value: 6

Enter a[1][2] value: 2

Enter a[2][0] value: 3

Enter a[2][1] value: 4

Enter a[2][2] value: 5

Input B matrix:

Enter b[0][0] value: 2

Enter b[0][1] value: 3

Enter b[0][2] value: 4

Enter b[1][0] value: 5

Enter b[1][1] value: 6

Enter b[1][2] value: 2

Enter b[2][0] value: 3

Enter b[2][1] value: 4

Enter b[2][2] value: 5

===== Matrix Multiplication Result =====

31 40 34
46 59 42
41 53 45

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

i) To insert a sub-string in to a given main string from a given position.

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

i) To insert a sub-string in to a given main string from a given position.

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

Program:

```
#include <stdio.h>

#include <string.h>

int main() {
   char str[200], sub[100], result[300];
   int pos;
   int i, j = 0;

printf("Enter the main string: ");
   gets(str);

printf("Enter the substring to insert: ");
   gets(sub);

printf("Enter position to insert: ");
   scanf("%d", &pos);

// Copy characters before the position
   for (i = 0; i < pos; i++)
          result[j++] = str[i];

// Insert substring
   for (i = 0; sub[i] != '\0'; i++)
          result[j++] = sub[i];
```

```
// Copy remaining characters of main string
   for (i = pos; str[i] != '\0'; i++)



result[j++] = str[i];

result[j] = '\0';

printf("\nString after insertion: %s\n", result);

   return 0;
   }
```

Output:

Enter the main string: Kg Reddy college

Enter the substring to insert: Autonomous

Enter position to insert: 9

String after insertion: Kg Reddy Autonomous college

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

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

Program:

```
#include <stdio.h>

#include <string.h>

int main() {
   char str[200], result[300];
   int pos, n;
   int i, j = 0;

printf("Enter the string: ");
   gets(str);

printf("Enter position to delete from: ");
   scanf("%d", &pos);

printf("Enter number of characters to delete: ");
   scanf("%d", &n);

// Copy characters before delete position
   for (i = 0; i < pos; i++)
          result[j++] = str[i];

// Skip n characters and copy the rest
   for (i = pos + n; str[i] != '\0'; i++)
          result[j++] = str[i];

result[j] = '\0';

printf("\nString after deletion: %s\n", result);

   return 0;}
```

Output:

Enter the string: kg reddy autonomous college

Enter position to delete from: 9

Enter number of characters to delete: 10

String after deletion: kg reddy college

12. Write a C program to determine if the given string is a palindrome or not

Aim: Write a C program to determine if the given string is a palindrome or not

Program:

```
#include<stdio.h>

#include<string.h>
   void main()
   {
   char str[10],strrev[10];
   int i,j,len;
   printf("Enter a string:");
   scanf("%s",str);
   len=strlen(str);
   for(i=len-1,j=0;i>=0;i--,j++)
   {
   strrev[j]=str[i];
   //printf("%c",str[i]);
   }
   strrev[j]='\0';
   if(strcmp(str,strrev)==0)
   printf("%s is palindrome",str);

else

printf("%s is not palindrome",str);
   }
```

Output:

```
   Enter a string: MADAM
   MADAM is palindrome
```

13. Write a C program that displays the position or index in the string S where the string T begins, or – 1 if S doesn't contain T.

Aim: A C program that displays the position or index in the string S where the string T begins, or – 1 if S doesn't contain T.

Program:

```
#include<stdio.h>
#include<string.h>
#include<conio.h>
void main()
  {
  char s[30];// t[20];
  char *found,t;
  /* Entering the main string */
  puts("Enter the first string: ");
  gets(s);
  /* Entering the string whose position or index to be displayed */
  puts("Enter the string to be searched: ");
  t=getchar();
  /*Searching string t in string s */
  // found=strstr(s,t);
  found =strchr(s,t);
  if(found)
  printf("Second String is found in the First String at %d position.\n",found-s);
else
printf("-1");
  }
```

Output:

Enter the first string:

kg reddy college

Enter the string to be searched:

college

Second String is found in the First String at 9 position.

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

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

Program:

```
#include <stdio.h>

#include<string.h>
   void main()
   {
   char line[81], ctr;
   int i,c,
   end = 0,
   characters = 0,
   words = 0,
   lines = 0;
   printf("KEY IN THE TEXT.\n");
   printf("GIVE ONE SPACE AFTER EACH WORD.\n");
   printf("WHEN COMPLETED, PRESS 'RETURN'.\n\n");
   while( end == 0)
   {
   /* Reading a line of text */

c = 0;

while((ctr=getchar()) != '\n')
   line[c++] = ctr;
   line[c] = '\0';
   /* counting the words in a line */
   if(line[0] == '\0')
   break ;

else
```

```
{
  words++;
  for(i=0; line[i] != '\0';i++)
  if(line[i] == ' ' || line[i] == '\t')
  words++;
  }
  /* counting lines and characters */
  lines = lines +1;
  characters = characters + strlen(line);
  }
  printf ("\n");
  printf("Number of lines = %d\n", lines);
  printf("Number of words = %d\n", words);
  printf("Number of characters = %d\n", characters);
  }
```

Output:

KEY IN THE TEXT.

GIVE ONE SPACE AFTER EACH WORD.

WHEN COMPLETED, PRESS 'RETURN'.

hello

I am Shoaib

Assistant professor

Number of lines = 3
  Number of words = 6
  Number of characters = 35

15. Write a C program to generate Pascal's triangle.

Aim: C program to generate Pascal's triangle.

Program:

```
#include <stdio.h>

int main() {
        int rows, i, j, num;

printf("Enter number of rows: ");
  scanf("%d", &rows);

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

num = 1;

// Print spaces for alignment
  for (j = 0; j < rows - i - 1; j++) {
        printf(" ");
  }

// Print numbers
  for (j = 0; j <= i; j++) {
        printf("%d ", num);
        num = num * (i - j) / (j + 1);
  }

  printf("\n");
  }

  return 0;
  }
```

Output:

Enter number of rows: 5

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

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

Aim: Program to construct a pyramid of numbers.

1

1 2

1 2 3

Program:

```
#include <stdio.h>

void main()
  {
  int i, j, rows;
  printf("Enter number of rows: ");
  scanf("%d",&rows);
  for(i=1; i<=rows; ++i)
  {
  for(j=1; j<=i; ++j)
  {
  printf("%d ",j);
  }
  printf("\n");
  }
  }
```

Output:

Enter number of rows: 3

1

1 2

1 2 3

ii)

Aim: Program to construct a pyramid of numbers.

1

2 3

4 5 6

Program:

```
#include <stdio.h>

void main()
  {
  int i, j, rows,num=1;
  printf("Enter number of rows: ");
  scanf("%d",&rows);
  for(i=1; i<=rows; ++i)
  {
  for(j=1; j<=i; ++j,num++)
  {
  printf("%d ",num);
  }
  printf("\n");
  }
  }
```

Output:

Enter number of rows: 3

1

2 3

4 5 6

iii) Aim: Program to construct a pyramid of numbers.

1

2 2

3 3 3

Program:

```
#include <stdio.h>

int main()
  {
        int i, j, rows;

printf("Enter number of rows: ");
  scanf("%d", &rows);

for (i = 1; i <= rows; i++)
  {
        for (j = 1; j <= i; j++)
        {
              printf("%d", i);
        }
        printf("\n");
  }

return 0; }
```

Output:

Enter number of rows: 3

1

22

333

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

i) Reading a complex number

ii) Writing a complex number

iii) Addition of two complex numbers

iv) Multiplication of two complex numbers

(Note: represent complex number using a structure.)

i) Reading a complex number

Aim: Reading a complex number

Program:

```
#include <stdio.h>

struct complex
  {
        float real;
        float imag;
  };

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

return c;

}

int main()
  {
        struct complex c1;
        c1 = readComplex();
```

```
printf("Complex number is: %.2f + %.2fi\n", c1.real, c1.imag);

  return 0;
  }
```

Output:

```
Enter real part: 3
  Enter imaginary part: 4
  Complex number is: 3.00 + 4.00i
```

ii) Writing a complex number

Aim: Writing a complex number

Program:

```
#include <stdio.h>

struct complex
  {
        float real;
        float imag;
  };

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

int main()
  {
        struct complex c1;

printf("Enter real part: ");
  scanf("%f", &c1.real);
  printf("Enter imaginary part: ");
  scanf("%f", &c1.imag);

writeComplex(c1);

  return 0;
  }
```

Output:

```
Enter real part: 5

Enter imaginary part: 2

Complex number is: 5.00 + 2.00i
```

iii) Addition of two complex numbers

Aim: Addition of complex number

Program:

```
#include <stdio.h>

struct complex
  {
        float real;
        float imag;
  };

struct complex addComplex(struct complex a, struct complex b)
  {
        struct complex sum;
        sum.real = a.real + b.real;
        sum.imag = a.imag + b.imag;

return sum;

}

int main()
  {
        struct complex c1, c2, result;

printf("Enter first complex number:\n");
  scanf("%f %f", &c1.real, &c1.imag);

printf("Enter second complex number:\n");
  scanf("%f %f", &c2.real, &c2.imag);

result = addComplex(c1, c2);

printf("Sum = %.2f + %.2fi\n", result.real, result.imag);
```

```
return 0;
}
```

Output:

```
Enter first complex number:
2 3
Enter second complex number:
4 5
Sum = 6.00 + 8.00i
```

iv) Multiplication of two complex numbers

Aim: Multiplication of complex number

Program:

```
#include <stdio.h>

struct complex
  {
        float real;
        float imag;
  };

struct complex multiplyComplex(struct complex a, struct complex b)
  {
        struct complex result;
        result.real = (a.real * b.real) - (a.imag * b.imag);
        result.imag = (a.real * b.imag) + (a.imag * b.real);
        return result;
  }

int main()
  {
        struct complex c1, c2, result;

printf("Enter first complex number:\n");
  scanf("%f %f", &c1.real, &c1.imag);

printf("Enter second complex number:\n");
  scanf("%f %f", &c2.real, &c2.imag);

result = multiplyComplex(c1, c2);

printf("Product = %.2f + %.2fi\n", result.real, result.imag);

  return 0;
  }
```

Output:

Enter first complex number:

1 2

Enter second complex number:

3 4

Product = -5.00 + 10.00i

18.

i. Write a C program which copies one file to another.

ii. Write a C program to reverse the first n characters in a file.
(Note: The file name and n are specified on the command line.)

i. Write a C program which copies one file to another.

Aim: To write a C program which copies one file to another.

Program:

```
#include <stdio.h>

int main()
  {
        FILE *fp1, *fp2;
        char ch;

/* Open source file in read mode */
  fp1 = fopen("source.txt", "r");

/* Open destination file in write mode */
  fp2 = fopen("destination.txt", "w");

/* Check if files opened successfully */
  if (fp1 == NULL || fp2 == NULL)
  {
        printf("File cannot be opened.\n");

  return 1;
  }

/* Copy character by character */
  while ((ch = fgetc(fp1)) != EOF)
  {
```

```
fputc(ch, fp2);

}

printf("File copied successfully.\n");

/* Close files */
   fclose(fp1);
   fclose(fp2);

   return 0;
   }
```

Output :

File copied successfully.

ii. Write a C program to reverse the first n characters in a file.

Aim: To write a C program to reverse the first n characters in a file.

Program:

```
#include <stdio.h>

#include <stdlib.h>

int main(int argc, char *argv[])
  {
        FILE *fp;
        char *buffer;
        int n, i;

/* Check command line arguments */
  if (argc != 3)
  {
        printf("Usage: program_name file_name n\n");

  return 1;
  }

/* Convert n from string to integer */
  n = atoi(argv[2]);

/* Open file in read and write mode */
  fp = fopen(argv[1], "r+");

if (fp == NULL)
  {
        printf("File cannot be opened.\n");

  return 1;
  }
```

```
/* Allocate memory */
   buffer = (char *)malloc(n * sizeof(char));

/* Read first n characters */
   fread(buffer, sizeof(char), n, fp);

/* Reverse the characters */
   for (i = 0; i < n / 2; i++)
   {
                    char temp = buffer[i];
          buffer[i] = buffer[n - i - 1];
          buffer[n - i - 1] = temp;
   }

/* Move file pointer to beginning */

fseek(fp, 0, SEEK_SET);

/* Write reversed characters back to file */

fwrite(buffer, sizeof(char), n, fp);

printf("First %d characters reversed successfully.\n", n);

/* Free memory and close file */

free(buffer);

fclose(fp);

return 0;

}
```

Output:

First 5 characters reversed successfully.

19.

i. Write a C program to display the contents of a file.

ii. 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)

i. Write a C program to display the contents of a file.

Aim: Write a C program to display the contents of a file.

Program:

```
#include <stdio.h>

#include <stdlib.h>

int main(int argc, char *argv[])
  {
        FILE *fs;
        int ch;

if (argc != 2)
  {
        printf("Invalid number of arguments.\n");

  return 1;
  }

fs = fopen(argv[1], "r");
  if (fs == NULL)
  {
        printf("Source file cannot be opened.\n");

  return 1;
  }
```

```
while ((ch = fgetc(fs)) != EOF)
{
        putchar(ch);
  }
  fclose(fs);
        return 0;
  }
```

Output:

Hey, how are you?

File handling in C is working fine

Thank you!!

ii. 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: To write a 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;
        char ch;

if (argc != 4)
  {
        printf("Usage: %s source1 source2 target\n", argv[0]);
                exit(1);
  }

f1 = fopen(argv[1], "r");
  if (f1 == NULL)
  {
        printf("Cannot open first source file.\n");
                exit(1);
  }

f2 = fopen(argv[2], "r");
  if (f2 == NULL)
  {
        printf("Cannot open second source file.\n");
        fclose(f1);
                exit(1);
  }
```

```
f3 = fopen(argv[3], "w");
  if (f3 == NULL)
  {
        printf("Cannot open target file.\n");
        fclose(f1);
        fclose(f2);
              exit(1);
  }

/* Copy first file */
  while ((ch = fgetc(f1)) != EOF)

{
        fputc(ch, f3);
  }

/* Copy second file */
        while ((ch = fgetc(f2)) != EOF)
        {
              fputc(ch, f3);
        }

fclose(f1);
  fclose(f2);
  fclose(f3);

printf("Files merged successfully.\n");

  return 0;
  }
```

Output:

Files merged successfully

20. Write a C program that uses functions to perform the following operations on singly linked list.:

i) Creation ii) Insertion iii) Deletion iv) Traversal

Aim: C program that uses functions to perform the following operations on singly linked list.:

i) Creation ii) Insertion iii) Deletion iv) Traversal

Program:

```
#include <stdio.h>

#include <stdlib.h>

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

struct node *head = NULL;

/* Create Linked List */
  void create() {
        int n, i;
        struct node *newnode, *temp;

printf("Enter number of nodes: ");

scanf("%d", &n);

for (i = 0; i < n; i++) {
        newnode = (struct node*)malloc(sizeof(struct node));

printf("Enter data: ");

scanf("%d", &newnode->data);
```

```
newnode->next = NULL;

        if (head == NULL) {
                        head = newnode;
                temp = head;
            } else {
                temp->next = newnode;
                temp = newnode;
            }
        }
   }

/* Insert at Beginning */
   void insert() {
        struct node *newnode;

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

printf("Enter data to insert: ");

scanf("%d", &newnode->data);

   newnode->next = head;
        head = newnode;
   }

/* Delete from Beginning */
   void delete() {
        struct node *temp;

if (head == NULL) {

printf("List is empty\n");

return;

}
```

```
temp = head;
   head = head->next;
   free(temp);

printf("Node deleted successfully\n");

}

/* Traverse Linked List */
   void traverse() {
            struct node *temp;

if (head == NULL) {

printf("List is empty\n");

return;

}

temp = head;

printf("Linked List: ");

while (temp != NULL) {

printf("%d -> ", temp->data);

   temp = temp->next;
   }

printf("NULL\n");

}

/* Main Function */

int main() {
            int choice;

while (1) {
```

```
printf("\n--- MENU ---\n");
   printf("1. Create\n");
   printf("2. Insert\n");
   printf("3. Delete\n");

printf("5. Exit\n");




printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {
   case 1: create(); break;
   case 2: insert(); break;
   case 3: delete(); break;
   case 4: traverse(); break;
   case 5: exit(0);

default: printf("Invalid choice\n");

            }
            }
            return 0;
   }
```

Output:

--- MENU ---

1. Create

2. Insert

3. Delete

4. Traverse

5. Exit

Enter your choice: 1

Enter number of nodes: 3

Enter data: 10

Enter data: 20

Enter data: 30

Enter your choice: 4

Linked List: 10 -> 20 -> 30 -> NULL

Enter your choice: 2

Enter data to insert: 5

Enter your choice: 4

Linked List: 5 -> 10 -> 20 -> 30 -> NULL

Enter your choice: 3

Node deleted successfully

Enter your choice: 4

Linked List: 10 -> 20 -> 30 ->NULL

21. Write C programs that implement stack (its operations) using

i) Arrays ii) Pointers

i) Arrays

Aim: C program that implement stack (its operations) using Arrays

Program:

```
#include <stdio.h>

#define MAX 5

int stack[MAX];

int top = -1;

/* Push operation */

void push() {

int x;

if (top == MAX - 1) {

printf("Stack Overflow\n");

return;

}

printf("Enter element to push: ");

scanf("%d", &x);

stack[++top] = x;

}
```

```
/* Pop operation */
void pop() {
if (top == -1) {
printf("Stack Underflow\n");
return;
}
printf("Popped element: %d\n", stack[top--]);
}
/* Display operation */
void display() {
int i;
if (top == -1) {
printf("Stack is empty\n");
return;
}
printf("Stack elements: ");
for (i = top; i >= 0; i--)
printf("%d ", stack[i]);
printf("\n");
}
```

```
/* Main function */

int main() {

int choice;

while (1) {

printf("\n--- STACK MENU (ARRAY) ---\n");

printf("1. Push\n");

printf("2. Pop\n");

printf("3. Display\n");

printf("4. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1: push(); break;

case 2: pop(); break;

case 3: display(); break;

case 4: return 0;

default: printf("Invalid choice\n");

}

}

}
```

Output:

--- STACK MENU (ARRAY) ---

1. Push

2. Pop

3. Display

4. Exit

Enter your choice: 1

Enter element to push: 10

--- STACK MENU (ARRAY) ---

1. Push

2. Pop

3. Display

4. Exit

Enter your choice: 1

Enter element to push: 20

--- STACK MENU (ARRAY) ---

1. Push

2. Pop

3. Display

4. Exit

Enter your choice: 3

Stack elements: 20 10

ii) C program that implement stack (its operations) using Pointers

Program:

```
#include <stdio.h>
#include <stdlib.h>
/* Stack node */
struct node {
int data;
struct node *next;
};
struct node *top = NULL;
/* Push operation */
void push() {
struct node *newnode;
newnode = (struct node*)malloc(sizeof(struct node));
printf("Enter element to push: ");
scanf("%d", &newnode->data);
newnode->next = top;
top = newnode;
}
```

```
/* Pop operation */

void pop() {

struct node *temp;

if (top == NULL) {

printf("Stack Underflow\n");

return;

}

temp = top;

printf("Popped element: %d\n", temp->data);

top = top->next;

free(temp);

}

/* Display operation */

void display() {

struct node *temp;

if (top == NULL) {

printf("Stack is empty\n");

return;

}

temp = top;

printf("Stack elements: ");
```

```
while (temp != NULL) {
printf("%d ", temp->data);
temp = temp->next;
}
printf("\n");
}
/* Main function */
int main() {
int choice;
while (1) {
printf("\n--- STACK MENU (POINTERS) ---\n");
printf("1. Push\n");
printf("2. Pop\n");
printf("3. Display\n");
printf("4. Exit\n");
printf("Enter your choice: ");
scanf("%d", &choice);
switch (choice) {
case 1: push(); break;
case 2: pop(); break;
case 3: display(); break;
```

```
case 4: return 0;

default: printf("Invalid choice\n");

}

}

}
```

Output:

--- STACK MENU (POINTERS) ---

1. Push

2. Pop

3. Display

4. Exit

Enter your choice: 1

Enter element to push: 5

--- STACK MENU (POINTERS) ---

1. Push

2. Pop

3. Display

4. Exit

Enter your choice: 3

Stack elements: 15 5

--- STACK MENU (POINTERS) ---

1. Push

2. Pop

3. Display

4. Exit

Enter your choice: 2

Popped element: 15

--- STACK MENU (POINTERS) ---

1. Push

2. Pop

3. Display

4. Exit

Enter your choice: 3

Stack elements: 5

22. Write C programs that implement Queue (its operations) using

i) Arrays ii) Pointers

i) Arrays

Aim: C program that implement Queue (its operations) using Arrays

Program:

```
#include <stdio.h>

#define MAX 5

int queue[MAX];

int front = -1, rear = -1;

/* Enqueue operation */

void enqueue() {

int x;

if (rear == MAX - 1) {

printf("Queue Overflow\n");

return;

}

printf("Enter element to insert: ");

scanf("%d", &x);

if (front == -1)

front = 0;

  queue[++rear] = x;
  }
```

```
/* Dequeue operation */

void dequeue() {
        if (front == -1 || front > rear) {

printf("Queue Underflow\n");

return;

}

printf("Deleted element: %d\n", queue[front++]);

}

/* Display operation */

void display() {

int i;

if (front == -1 || front > rear) {

printf("Queue is empty\n");

return;

}

printf("Queue elements: ");

for (i = front; i <= rear; i++)

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



printf("\n");

}
```

```
/* Main function */

int main() {
        int choice;

while (1) {

printf("\n--- QUEUE MENU (ARRAY) ---\n");
   printf("1. Enqueue\n");
   printf("2. Dequeue\n");
   printf("3. Display\n");
   printf("4. Exit\n");
   printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {
   case 1: enqueue(); break;
   case 2: dequeue(); break;
   case 3: display(); break;

case 4: return 0;

default: printf("Invalid choice\n");

        }
        }
   }
```

Output:

```
--- QUEUE MENU (ARRAY) ---
   1. Enqueue
   2. Dequeue
   3. Display

   4. Exit
```

Enter your choice: 1

Enter element to insert: 10

Enter your choice: 1

Enter element to insert: 20

Enter your choice: 3

Queue elements: 10 20

Enter your choice: 2

Deleted element: 10

Enter your choice: 3

Queue elements: 20

ii) C program that implement Queue (its operations) using Pointers

Am: C program that implement Queue (its operations) using Pointers

Program:

```
#include <stdio.h>
#include <stdlib.h>
struct node {
int data;
struct node *next;
};
struct node *front = NULL;
struct node *rear = NULL;
/* Enqueue operation */
void enqueue() {
struct node *newnode;
newnode = (struct node*)malloc(sizeof(struct node));
printf("Enter element to insert: ");
scanf("%d", &newnode->data);
newnode->next = NULL;
if (rear == NULL) {
front = rear = newnode;
} else {
```

```
rear->next = newnode;

rear = newnode;

}

}

/* Dequeue operation */

void dequeue() {

struct node *temp;

if (front == NULL) {

printf("Queue Underflow\n");

return;

}

temp = front;

printf("Deleted element: %d\n", temp->data);

front = front->next;

if (front == NULL)

rear = NULL;

free(temp);

}

/* Display operation */

void display() {

struct node *temp;
```

```
if (front == NULL) {
printf("Queue is empty\n");
return;
}
temp = front;
printf("Queue elements: ");
while (temp != NULL) {
printf("%d ", temp->data);
temp = temp->next;
}
printf("\n");
}
/* Main function */
int main() {
int choice;
while (1) {
printf("\n--- QUEUE MENU (POINTERS) ---\n");
printf("1. Enqueue\n");
printf("2. Dequeue\n");
printf("3. Display\n");
printf("4. Exit\n");
printf("Enter your choice: ");
```

```
scanf("%d", &choice);

switch (choice) {

case 1: enqueue(); break;

case 2: dequeue(); break;

case 3: display(); break;

case 4: return 0;

default: printf("Invalid choice\n");

}

}

}
```

Output:

--- QUEUE MENU (POINTERS) ---

1. Enqueue

2. Dequeue

3. Display

4. Exit

Enter your choice: 1

Enter element to insert: 5

Enter your choice: 1

Enter element to insert: 15

Enter your choice: 3

Queue elements: 5 15

Enter your choice: 2

Deleted element: 5

Enter your choice: 3

Queue elements: 15

23. Write a C program that implements the following sorting methods to sort a given list of integers in ascending order i) Bubble sort ii) Selection sort iii)Insertion sort

i) Bubble Sort

Aim: 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 integers */

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 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 elements */

void printlist(int list[], int n) {
   int j;
   for (j = 0; j < n; j++)

   printf("%d\t", list[j]);
   printf("\n");

}

/* Main function */

int main() {

int list[MAX], num;

printf("\n***** Enter the number of elements [Maximum 10] *****\n");

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("\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:

5 2 4 1 3

Elements in the list before sorting are:

5 2 4 1 3

Elements in the list after sorting are:

1 2 3 4 5
```

ii)Selection Sort

Aim: Program that implements the Selection sort method to sort a given list of integers in ascending order

Program:

```
#include <stdio.h>

#define MAX 10

/* Function to swap two integers */

void swapList(int *m, int *n) {

int temp = *m;

*m = *n;

*n = temp;

}

/* Function for Selection Sort */

void sel_sort(int list[], int n) {

int i, j, min;

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

min = i; // Assume current position has minimum

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

if (list[j] < list[min])

min = j; // Update index of minimum

}

if (min != i)
```

```
swapList(&list[i], &list[min]); // Swap with current element
}
}
/* Function to read 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 elements */
void printlist(int list[], int n) {
int j;
for (j = 0; j < n; j++)
printf("%d\t", list[j]);
printf("\n");
}
/* Main function */
int main() {
int list[MAX], num;
```

```
printf("\n***** Enter the number of elements [Maximum 10] *****\n");

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);

sel_sort(list, num);

printf("\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:

29 10 14 37 13

Elements in the list before sorting are:

29 10 14 37 13

Elements in the list after sorting are:

10 13 14 29 37

iii) Insertion sort

Aim: Program that implements the Insertion sort method to sort a given list of integers in ascending order

Program:

```
#include <stdio.h>

#define MAX 64

int main() {

int n, i, j, temp;

int arr[MAX];

printf("Enter number of elements: ");

scanf("%d", &n);

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

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

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

// Insertion Sort

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

Output:

```
Enter number of elements: 5
Enter 5 integers:
25 12 37 4 18
Sorted list in ascending order:
4
12
18
25
37
```

24. Write C programs that use both recursive and non recursive functions to perform the following searching operations for a Key value in a given list of integers:

i) Linear search ii) Binary search

i) Linear Search

Aim: C programs that use both recursive and non recursive functions to perform the Linear search

```
#include <stdio.h>

/* Non-Recursive Linear Search */

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

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

if (arr[i] == key)

return i; // index found

}

return -1; // not found

}

/* Recursive Linear Search */

int linear_recursive(int arr[], int n, int key, int index) {

if (index >= n)

return -1;

if (arr[index] == key)

return index;

return linear_recursive(arr, n, key, index + 1);
```

```
}

int main() {

int n, arr[50], key, index;

printf("Enter number of elements: ");

scanf("%d", &n);

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

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

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

printf("Enter key to search: ");

scanf("%d", &key);

// Non-Recursive Linear Search

index = linear_nonrecursive(arr, n, key);

if (index != -1)

printf("Key found at position (Non-Recursive Linear Search): %d\n", index + 1);

else

printf("Key not found (Non-Recursive Linear Search)\n");

// Recursive Linear Search

index = linear_recursive(arr, n, key, 0);

if (index != -1)

printf("Key found at position (Recursive Linear Search): %d\n", index + 1);

else
```

```
printf("Key not found (Recursive Linear Search)\n");


return 0;

}
```

Output:

```
Enter number of elements: 5

Enter 5 integers:

10 20 30 40 50

Enter key to search: 30

Key found at position (Non-Recursive Linear Search): 3

Key found at position (Recursive Linear Search): 3
```

ii) **Binary Search**

Aim: C programs that use both recursive and non recursive functions to perform the Binary search.

Program:

```
#include <stdio.h>

/* Non-Recursive Binary Search */

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

int low = 0, high = n - 1, mid;

while (low <= high) {

mid = (low + high) / 2;

if (arr[mid] == key)

return mid;

else if (arr[mid] < key)

low = mid + 1;

else

high = mid - 1;

}

return -1; // not found

}

/* Recursive Binary Search */

int binary_recursive(int arr[], int low, int high, int key) {

if (low > high)
```

```
return -1;
int mid = (low + high) / 2;
if (arr[mid] == key)
return mid;
else if (arr[mid] < key)
return binary_recursive(arr, mid + 1, high, key);
else
return binary_recursive(arr, low, mid - 1, key);
}
int main() {
int n, arr[50], key, index;
printf("Enter number of elements: ");
scanf("%d", &n);
printf("Enter %d sorted integers:\n", n);
for (int i = 0; i < n; i++)
scanf("%d", &arr[i]);
printf("Enter key to search: ");
scanf("%d", &key);
// Non-Recursive Binary Search
index = binary_nonrecursive(arr, n, key);
if (index != -1)
```

```
printf("Key found at position (Non-Recursive Binary Search): %d\n", index + 1);

else

printf("Key not found (Non-Recursive Binary Search)\n");

// Recursive Binary Search

index = binary_recursive(arr, 0, n - 1, key);

if (index != -1)

printf("Key found at position (Recursive Binary Search): %d\n", index + 1);

else

printf("Key not found (Recursive Binary Search)\n");

return 0;

}
```

Output:

```
Enter number of elements: 6

Enter 6 sorted integers:

5 10 15 20 25 30

Enter key to search: 20

Key found at position (Non-Recursive Binary Search): 4

Key found at position (Recursive Binary Search): 4
```