- What is an algorithm?
- What are the qualities of a good algorithm?
- Write an algorithm to:
  - Add two numbers entered by the user
  - Find the largest number among two numbers
  - Find the largest number among three distinct numbers
  - Find roots of a quadratic equation ($ax^2 + bx + c = 0$)
  - Display all natural numbers from 1 to 10
  - Display all natural numbers from 100 to 1
  - Display all even natural numbers from 100 to 1

- What is a flowchart?
- What are the various symbols used in a flowchart?
- Draw flowcharts for each of the above mentioned algorithms.
- What are keywords? Enlist the various keywords of C language.
- What are identifiers? What are the rules for naming?
- What are data types? What size does each have?

- Write a program in C to implement the following:

1. Display your name
2. Add two numbers and display the result
3. Read two numbers from keyboard and compare them
4. Swap two numbers
5. Define and use a constant (e.g. PI as 3.14159) in a program
6. Use *math* library (pow(), sqrt(), etc.)
7. Use *if-else* to check whether a number is even or odd
8. Find roots of a quadratic equation
9. Use ternary operator
10. Use *nesting* of *if-else* (compare three numbers)

- Study various operators used in C language: Relational Logical, Bitwise, Assignment operators.
- Study precedence of operators.

11. Use if-else with relational and logical operators (grading according to percentage of a student).
12. Use *switch-case* to display Salaam when user enters 1, Aadaab when user enters 2, Hello when user enters 3 and Incorrect Option when user enters any other number

13. Display the series (using loops): 1 2 3 4 5 .... n

14. Display the series (using loops): n n-1 n-2 …. 3 2 1

15. Display all even numbers from 1 to 100 [using loop and if]

16. Find sum of series: 1 2 3 4 5 .... n

17. Find product of series: 1 2 3 4 5 .... n

18. Find factorial of a number entered by the user

19. Find all the factors of a natural number

20. Check whether a number is prime or not

21. Display the series: 2 4 8 16 32 64 128 ..... (n terms)

22. Display the series: 2 4 16 256 65536 ..... (n terms)

23. Display the Fibonacci series: 0 1 1 2 3 5 8 13 21 ..... (n terms)

24. Display table of all integers from 2 to 10 (*2x1=2, 2x2=4, ……, 10x10=100)*

25. Display the series: 1 2 2 3 3 3 4 4 4 4 ..... (up to n) [nesting of loops]

26. Display the series: 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 ..... (n times) [using nesting of loops]

27. Find sum of series: 1 + 1/2 + 1/4 + 1/6 + ….+ 1/n

28. Display the digits of any natural number in reverse order [Use while loop]

29. Check whether a number is Armstrong number or not

30. Demonstrate the use of *continue* and *break* keywords (in loops)

31. Display the following pattern (using loops):

```
*
**
***
****
```

32. Display the following pattern (using loops):

```
1
12
123
1234
```

33. Display the following pattern (using loops):

```
**
***
****
*****
```

34. Display the following pattern (using loops):

```
*****
***
**
*
```

35. Display the following pattern (using loops):

```
    *
   **
  ***
 ****
```

36. Display the following pyramid of stars (using loops):

```
    *
   * *
  * * *
 * * * *
```

37. Display contents of an integer array.

38. Input an integer array from the user and display the same.

39. Display contents of an integer array in reverse order.

40. Display contents of a character array.

41. Input a string from the user and display the same.

42. Input a string from the user and find its length (without using the *string* library).

43. Input a string from the user and display it in reverse order.

44. Check whether a string is palindrome or not.

45. Demonstrate the use of s*tring* library functions: strlen(), strcpy(), strrev(), strcat(), strcmp() etc. using examples.

46. Find sum of all elements of an integer array.

47. Search for an element in an integer array (*Linear search*).

48. Sort an integer array using *Bubble Sort*.

49. Input a *matrix* from the user and display it.

50. Find transpose of a matrix.

51. Find the sum of all diagonal elements of a matrix.

52. Find sum of two matrices.

53. Find product of two matrices.

54. Swap two numbers using *functions* (call-by-value method).

55. Add and subtract two integers using functions (a+b, a-b, b-a).

56. Search an array using functions (passing array to a function).

57. Find factorial of a number using *recursion*.

58. Implement Ackermann Function using recursion.

59. Display Fibonacci series up to n terms using recursion.

60. Demonstrate use of *macros*.

61. Store details of a student in a structure and then display the same.

62. Store details of all the students of your class in an array of structure and then display the same in a tabular format.

63. Store details (Roll No. and Marks) of all the students of your class in an array of structure and then find the topper of the class.

64. Demonstrate use of *pointers* in programming.

65. Swap two numbers using functions (call-by-reference method).

66. Display contents of an array using pointers.

67. Display contents of a string in reverse order using pointers.

68. Check if a string is palindrome or not using pointers.

69. Sort an integer array using functions and pointers.

70. Demonstrate use of pointers in self-referential structures (theory).

71. Describe the various storage classes in C (theory).

72. Read contents of a text *file*.

73. Write into a text file.

74. Find the number of characters, words, sentences, vowels, consonants, punctuation marks, etc. in a text file.