**Bitwise operators**:

1. Write a C program to check Least Significant Bit (LSB) of a number is set or not.

2. Write a C program to check Most Significant Bit (MSB) of a number is set or not.

3. Write a C program to get nth bit of a number.

4. Write a C program to set nth bit of a number.

5. Write a C program to clear nth bit of a number.

6. Write a C program to toggle nth bit of a number.

7. Write a C program to get highest set bit of a number.

8. Write a C program to get lowest set bit of a number.

9. Write a C program to count trailing zeros in a binary number.

10. Write a C program to count leading zeros in a binary number.

11. Write a C program to flip bits of a binary number using bitwise operator.

12. Write a C program to count total zeros and ones in a binary number.

13. Write a C program to rotate bits of a given number.

14. Write a C program to convert decimal to binary number system using bitwise operator.

15. Write a C program to swap two numbers using bitwise operator.

16. Write a C program to check whether a number is even or odd using bitwise operator.

17. set 2 bits of number given pos1, pos2 and make the bits in those positions as zero

18. Write a C program to swap even and odd bits.

19. Write a C program to check if the given number is power of 2.

20. Implement own sizeof operator

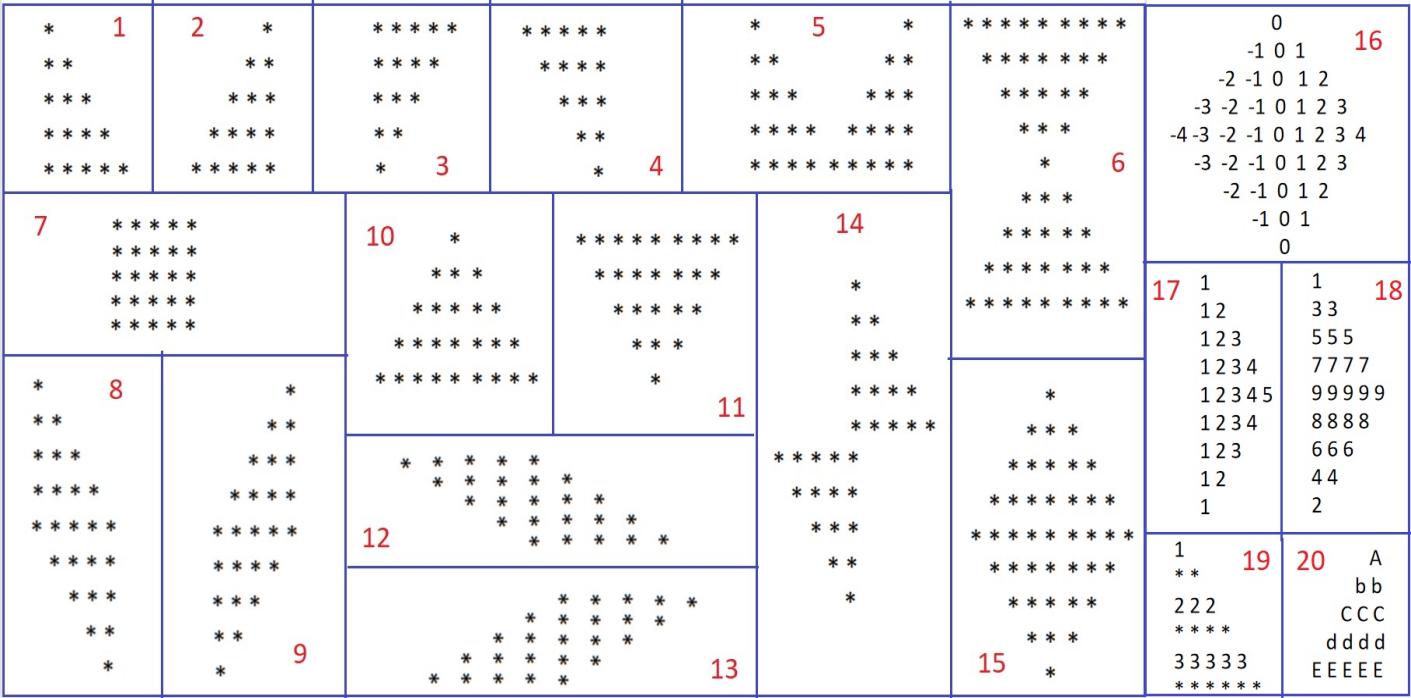
21. Implement own memcpy

22. Write a function to calculate average of given numbers in an array

23. WAP to swap alt bits of a char i/p: 10 01 01 01 o/p: 01 01 10 01

**Pattern programs (Loops):**

Write programs for all the patterns mentioned in the below image.



**Pointer programs**:

1. Write a C program to create, initialize and use pointers.

2. Write a C program to add two numbers using pointers.

3. Write a C program to swap two numbers using pointers.

4. Write a C program to input and print array elements using pointer.

5. Write a C program to copy one array to another using pointers.

6. Write a C program to swap two arrays using pointers.

7. Write a C program to reverse an array using pointers.

8. Write a C program to search an element in array using pointers.

9. Write a C program to access two dimensional array using pointers.

10. Write a C program to add two matrix using pointers.

11. Write a C program to multiply two matrix using pointers.

12. Write a C program to find length of string using pointers.

13. Write a C program to copy one string to another using pointers.

14. Write a C program to concatenate two strings using pointers.

15. Write a C program to compare two strings using pointers.

16. Write a C program to find reverse of a string using pointers.

17. Write a C program to sort array using pointers.

18. Write a C program to return multiple value from function using pointers.

**Recursive Functions**:

1. Write a C program to print all natural numbers between 1 to n using recursion.

2. Write a C program to print all even or odd numbers in given range using recursion.

3. Write a C program to find sum of all natural numbers between 1 to n using recursion.

4. Write a C program to find sum of all even or odd numbers in given range using recursion.

5. Write a C program to find reverse of any number using recursion.

6. Write a C program to check whether a number is palindrome or not using recursion.

7. Write a C program to find sum of digits of a given number using recursion.

8. Write a C program to find factorial of any number using recursion.

**String Programs**:

1. Write a C program to find length of a string.

2. Write a C program to copy one string to another string.

3. Write a C program to concatenate two strings.

4. Write a C program to compare two strings.

5. Write a C program to find total number of alphabets, digits or special character in a string.

6. Write a C program to count total number of vowels and consonants in a string.

7. Write a C program to count total number of words in a string.

8. Write a C program to find reverse of a string.

9. Write a C program to check whether a string is palindrome or not.

10. Write a C program to find first occurrence of a character in a given string.

11. Write a C program to count occurrences of a character in given string.

12. Write a C program to find highest frequency character in a string.

13. Write a C program to count frequency of each character in a string.

14. Write a C program to remove all occurrences of a character from string.

15. Write a C program to remove all repeated characters from a given string.

16. Write a C program to replace all occurrences of a character with another in a string.

17. Write a C program to find first occurrence of a word in a given string.

**Array Programs**:

1. Write a C program to find sum of all array elements. - using recursion.

2. Write a C program to find maximum and minimum element in an array. - using recursion.

3. Write a C program to find second largest element in an array.

4. Write a C program to count total number of even and odd elements in an array.

5. Write a C program to count total number of negative elements in an array.

6. Write a C program to copy all elements from an array to another array.

7. Write a C program to insert an element in an array.

8. Write a C program to delete an element from an array at specified position.

9. Write a C program to count frequency of each element in an array.

10. Write a C program to print all unique elements in the array.

11. Write a C program to count total number of duplicate elements in an array.

12. Write a C program to delete all duplicate elements from an array.

13. Write a C program to merge two arrays to third array.

**File Handling Programs**:

1. Write a C program to create a file and write contents, save and close the file.

2. Write a C program to read file contents and display on console.

3. Write a C program to append content to a file.

4. Write a C program to count characters, words and lines in a text file.

5. Write a C program to compare two files.

6. Write a C program to copy contents from one file to another file.

**DS**:

1. Sorting algorithms - Bubble sort, Quick sort, Selection sort and insertion sort --- Time complexity and applications

2. Deletion of a last node in circular linked list

3. Deletion of alternative node in single linked list

4. Creating a new node in circular Linked list at nth position

5. Implement stack using queues and queues using stacks

6. Count no of nodes in circular LL

7. Reverse a SLL

**Threads**:

1. Print odd and even numbers using two threads in C