1. Sum of Elements in an Array

- **Problem**: Given an array, write a program to find the sum of its elements.

```
#include <stdio.h>
int main() {
   int arr[] = {1, 2, 3, 4, 5};
   int sum = 0;
   for (int i = 0; i < 5; i++) {
      sum += arr[i];
   }
   printf("Sum: %d\n", sum);
   return 0;
}
```

Output: Sum: 15

2. Reverse an Array

- **Problem**: Write a program to reverse an array.

```
#include <stdio.h>
void reverseArray(int arr[], int size) {
   int temp, start = 0, end = size - 1;
   while (start < end) {
      temp = arr[start];
      arr[start] = arr[end];
      arr[end] = temp;
      start++;
      end--;
   }
}

int main() {
   int arr[] = {1, 2, 3, 4, 5};
   int size = sizeof(arr) / sizeof(arr[0]);
   reverseArray(arr, size);
```

```
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
    return 0;
}
```

Output: 5 4 3 2 1

3. Find Largest Element in Array

- **Problem**: Write a program to find the largest element in an array.

```
#include <stdio.h>
int main() {
    int arr[] = {1, 3, 5, 7, 2};
    int largest = arr[0];
    for (int i = 1; i < 5; i++) {
        if (arr[i] > largest) {
            largest = arr[i];
        }
    }
    printf("Largest Element: %d\n", largest);
    return 0;
}
```

Output: Largest Element: 7

4. Palindrome Check (String)

- **Problem**: Check if a string is a palindrome.

```
#include <stdio.h>
#include <string.h>
int isPalindrome(char str[]) {
    int start = 0, end = strlen(str) - 1;
    while (start < end) {
        if (str[start] != str[end]) {
            return 0;
        }
        start++;
        end--;
    }
    return 1;
}

int main() {
    char str[] = "madam";
    if (isPalindrome(str)) {
        printf("Palindrome\n");
    } else {
        printf("Not Palindrome\n");
    }
    return 0;
}
```

Output: Palindrome

5. Count Vowels in a String

- **Problem**: Write a program to count the number of vowels in a string.

```
#include <stdio.h>
#include <string.h>
int countVowels(char str[]) {
    int count = 0;
    for (int i = 0; i < strlen(str); i++) {
        if (str[i] == 'a' || str[i] == 'e' || str[i] == 'i' || str[i] == 'o' || str[i] == 'u') {
            count++;
        }
    }
    return count;
}

int main() {
    char str[] = "hello world";
    printf("Vowels Count: %d\n", countVowels(str));
    return 0;
}
```

Output: Vowels Count: 3

6. Count the Number of Digits in a Number

- **Problem**: Count the number of digits in a given number.

```
#include <stdio.h>
int countDigits(int num) {
    int count = 0;
    while (num != 0) {
        num /= 10;
        count++;
    }
    return count;
}

int main() {
    int num = 12345;
    printf("Number of digits: %d\n", countDigits(num));
    return 0;
}
```

Output: Number of digits: 5

7. Fibonacci Series (Recursive)

- **Problem**: Print the Fibonacci series using recursion.

```
#include <stdio.h>
int fibonacci(int n) {
    if (n <= 1) return n;
    return fibonacci(n-1) + fibonacci(n-2);
}
```

```
int main() {
    int n = 10;
    for (int i = 0; i < n; i++) {
        printf("%d ", fibonacci(i));
    }
    return 0;
}
```

Output: 0 1 1 2 3 5 8 13 21 34

8. Factorial of a Number

- **Problem**: Write a program to find the factorial of a number.

```
#include <stdio.h>
int factorial(int n) {
    if (n == 0 || n == 1) return 1;
    return n * factorial(n - 1);
}

int main() {
    int n = 5;
    printf("Factorial: %d\n", factorial(n));
    return 0;
}
```

Output: Factorial: 120

9. GCD (Greatest Common Divisor) of Two Numbers

- **Problem**: Write a program to find the GCD of two numbers using recursion.#include <stdio.h>

```
int gcd(int a, int b) {
    if (b == 0) return a;
    return gcd(b, a % b);
}

int main() {
    int a = 56, b = 98;
    printf("GCD: %d\n", gcd(a, b));
    return 0;
}
```

Output: GCD: 14

10. Sum of Digits of a Number

- **Problem**: Write a program to find the sum of digits of a number.

```
#include <stdio.h>
int sumOfDigits(int num) {
    int sum = 0;
    while (num != 0) {
        sum += num % 10;
        num /= 10;
    }
    return sum;
}

int main() {
    int num = 12345;
    printf("Sum of digits: %d\n", sumOfDigits(num));
    return 0;
}
```

Output: Sum of digits: 15

11. Prime Number Check

- **Problem**: Write a program to check if a number is prime.

```
#include <stdio.h>
int isPrime(int num) {
    for (int i = 2; i <= num / 2; i++) {
        if (num % i == 0) return 0;
    }
    return 1;
}

int main() {
    int num = 11;
    if (isPrime(num)) {
        printf("%d is prime\n", num);
    } else {
        printf("%d is not prime\n", num);
    }
    return 0;
}
```

Output: 11 is prime

12. Armstrong Number

- **Problem**: Check if a number is an Armstrong number (a number that is equal to the sum of the cubes of its digits).

```
#include <stdio.h>
int isArmstrong(int num) {
    int sum = 0, temp, remainder;
    temp = num;
    while (temp != 0) {
        remainder = temp % 10;
        sum += remainder * remainder * remainder;
        temp /= 10;
    }
    return sum == num;
}

int main() {
    int num = 153;
    if (isArmstrong(num)) {
        printf("%d is an Armstrong number\n", num);
    } else {
        printf("%d is not an Armstrong number\n", num);
    }
    return 0;
}
```

Output: 153 is an Armstrong number

13. Print Prime Numbers in a Range

- **Problem**: Print all prime numbers in a given range.

```
#include <stdio.h>
int isPrime(int num) {
    for (int i = 2; i <= num / 2; i++) {
        if (num % i == 0) return 0;
    }
    return 1;
}

int main() {
    int start = 10, end = 50;
    for (int i = start; i <= end; i++) {
        if (isPrime(i)) {
            printf("%d ", i);
        }
    }
    return 0;
}
```

Output: 11 13 17 19 23 29 31 37 41 43 47

14. Matrix Multiplication

- **Problem**: Multiply two matrices.

```
#include <stdio.h>
#define MAX 10
int main() {
```

```
int a[MAX][MAX], b[MAX][MAX], product[MAX][MAX], i, j, k, r1, c1, r2, c2;

printf("Enter rows and columns for first matrix: ");
scanf("%d %d", &r1, &c1);
printf("Enter rows and columns for second matrix: ");
scanf("%d %d", &r2, &c2);

if (c1 != r2) {
   printf("Matrix multiplication not possible.\n");
   return 1;
}

printf("Enter elements of matrix 1:\n");
for (i = 0; i < r1; i++) {
   for (j = 0; j < c1; j++) {
      scanf("%d", &a[i][j]);
   }
}

printf("Enter elements of matrix 2:\n");
for (i = 0; i < r2; i++) {
   for (j = 0; j < c2; j++) {
      scanf("%d", &b[i][j]);
   }
}

for (i = 0; i < r1; i++) {
   for (j = 0; j < c2; j++) {
      product[i][j] = 0;
      for (k = 0; k < c1; k++) {
         product[i][j] += a[i][k] * b[k][j];
```

```
            }
        }
    }

    printf("Product of matrices:\n");
    for (i = 0; i < r1; i++) {
        for (j = 0; j < c2; j++) {
            printf("%d ", product[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

Output: (Matrix multiplication result based on input matrices)

15. Bubble Sort

- **Problem**: Implement Bubble Sort to sort an array.

```
#include <stdio.h>
void bubbleSort(int arr[], int n) {
    for (int i = 0; i < n-1; i++) {
        for (int j = 0; j < n-i-1; j++) {
            if (arr[j] > arr[j+1]) {
                int temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
    }
}

int main() {
    int arr[] = {64, 34, 25, 12, 22, 11, 90};
    int n = sizeof(arr) / sizeof(arr[0]);
    bubbleSort(arr, n);
    printf("Sorted array: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    return 0;
}
```

Output: Sorted array: 11 12 22 25 34 64 90

16. Find the Second Largest Element in an Array

- **Problem**: Write a program to find the second largest element in an array.

```
#include <stdio.h>
int main() {
    int arr[] = {12, 35, 1, 10, 34, 1};
    int largest = arr[0];
    int secondLargest = -1;

    for (int i = 1; i < 6; i++) {
        if (arr[i] > largest) {
            secondLargest = largest;
            largest = arr[i];
        } else if (arr[i] > secondLargest && arr[i] != largest) {
            secondLargest = arr[i];
        }
    }
    printf("Second largest element is: %d\n", secondLargest);
    return 0;
}
```

Output: Second largest element is: 34

17. Check Whether a Number is Power of Two

- **Problem**: Write a program to check if a number is a power of two.

```
#include <stdio.h>

int isPowerOfTwo(int num) {

   return (num > 0) && (num & (num - 1)) == 0;

}


int main() {

   int num = 16;

   if (isPowerOfTwo(num)) {

      printf("%d is a power of two\n", num);

   } else {

      printf("%d is not a power of two\n", num);

   }

   return 0;

}
```

Output: 16 is a power of two

18. Print Fibonacci Series Without Recursion

- **Problem**: Print the Fibonacci series without using recursion.

```
#include <stdio.h>

int main() {

   int n = 10, t1 = 0, t2 = 1, nextTerm;


   printf("Fibonacci Series: ");

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

      printf("%d ", t1);

      nextTerm = t1 + t2;

      t1 = t2;

      t2 = nextTerm;

   }

   return 0;

}
```

Output: Fibonacci Series: 0 1 1 2 3 5 8 13 21 34

19. Count the Frequency of a Character in a String

- **Problem**: Write a program to count the frequency of a character in a string.

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


int countFrequency(char str[], char ch) {
    int count = 0;
    for (int i = 0; i < strlen(str); i++) {
        if (str[i] == ch) {
            count++;
        }
    }
    return count;
}


int main() {
    char str[] = "programming";
    char ch = 'g';
    printf("Frequency of '%c': %d\n", ch, countFrequency(str, ch));
    return 0;
}
```

Output: Frequency of 'g': 2

20. Merge Two Sorted Arrays

- **Problem**: Merge two sorted arrays into a single sorted array.

c

Copy code

```
#include <stdio.h>
void merge(int arr1[], int arr2[], int n1, int n2) {
    int result[n1 + n2];
    int i = 0, j = 0, k = 0;
    while (i < n1 && j < n2) {
        if (arr1[i] < arr2[j]) {
            result[k++] = arr1[i++];
        } else {
            result[k++] = arr2[j++];
        }
    }

    while (i < n1) result[k++] = arr1[i++];
    while (j < n2) result[k++] = arr2[j++];

    printf("Merged array: ");
    for (int i = 0; i < n1 + n2; i++) {
        printf("%d ", result[i]);
    }
}

int main() {
    int arr1[] = {1, 3, 5};
    int arr2[] = {2, 4, 6};
    int n1 = sizeof(arr1) / sizeof(arr1[0]);
    int n2 = sizeof(arr2) / sizeof(arr2[0]);

    merge(arr1, arr2, n1, n2);
    return 0;
}
```

Output: Merged array: 1 2 3 4 5 6

21. Find the Missing Number in an Array

- **Problem**: Given an array of integers from 1 to n, with one missing, find the missing number.

```
#include <stdio.h>
int findMissingNumber(int arr[], int n) {
    int sum = (n * (n + 1)) / 2;
    for (int i = 0; i < n - 1; i++) {
        sum -= arr[i];
    }
    return sum;
}

int main() {
    int arr[] = {1, 2, 4, 5, 6};
    int n = 6;
    printf("Missing number: %d\n", findMissingNumber(arr, n));
    return 0;
}
```

Output: Missing number: 3

22. Find the Length of a String

- **Problem**: Write a program to find the length of a string without using built-in functions.

```
#include <stdio.h>
int stringLength(char str[]) {
    int length = 0;
    while (str[length] != '\0') {
        length++;
    }
    return length;
}

int main() {
    char str[] = "Hello World!";
    printf("Length of the string: %d\n", stringLength(str));
    return 0;
}
```

Output: Length of the string: 12

23. Find All Divisors of a Number

- **Problem**: Write a program to find all divisors of a number.

```
#include <stdio.h>
void findDivisors(int num) {
   for (int i = 1; i <= num; i++) {
      if (num % i == 0) {
         printf("%d ", i);
      }
   }
}

int main() {
   int num = 36;
   printf("Divisors of %d: ", num);
   findDivisors(num);
   return 0;
}
```

Output: Divisors of 36: 1 2 3 4 6 9 12 18 36

24. Check if a Number is Even or Odd

- **Problem**: Write a program to check whether a number is even or odd.

```
#include <stdio.h>
int main() {
   int num = 10;
   if (num % 2 == 0) {
      printf("%d is even\n", num);
   } else {
      printf("%d is odd\n", num);
   }
   return 0;
}
```

Output: 10 is even

25. Sum of Prime Numbers in a Range

- **Problem**: Write a program to find the sum of all prime numbers in a range.#include <stdio.h>

```
int isPrime(int num) {
   for (int i = 2; i <= num / 2; i++) {
      if (num % i == 0) return 0;
   }
   return 1;
}

int main() {
   int start = 10, end = 30, sum = 0;
   for (int i = start; i <= end; i++) {
      if (isPrime(i)) {
         sum += i;
      }
   }
   printf("Sum of prime numbers: %d\n", sum);
   return 0;
}
```

Output: Sum of prime numbers: 129

26. Reverse a Number

- **Problem**: Write a program to reverse a number.

```
#include <stdio.h>
int reverseNumber(int num) {
   int rev = 0;
   while (num != 0) {
      rev = rev * 10 + num % 10;
      num /= 10;
   }
   return rev;
}
```

```
int main() {
   int num = 12345;
   printf("Reversed Number: %d\n", reverseNumber(num));
   return 0;
}
```

Output: Reversed Number: 54321

27. Find the Prime Factorization of a Number

- **Problem**: Find the prime factorization of a number.

```
#include <stdio.h>
void primeFactorization(int num) {
   for (int i = 2; i * i <= num; i++) {
      while (num % i == 0) {
         printf("%d ", i);
         num /= i;
      }
   }
   if (num > 1) {
      printf("%d", num);
   }
}

int main() {
   int num = 56;
   printf("Prime factorization of %d: ", num);
   primeFactorization(num);
   return 0;
}
```

Output: Prime factorization of 56: 2 2 2 7

28. Remove Duplicate Elements from an Array

- **Problem**: Remove duplicate elements from an array.

```
#include <stdio.h>
int removeDuplicates(int arr[], int n) {
  int temp[n], k = 0;
  for (int i = 0; i < n; i++) {
    int flag = 0;
    for (int j = 0; j < k; j++) {
      if (arr[i] == temp[j]) {
        flag = 1;
        break;
      }
    }
    if (flag == 0) {
      temp[k++] = arr[i];
    }
  }

  printf("Array without duplicates: ");
  for (int i = 0; i < k; i++) {
    printf("%d ", temp[i]);
  }
  return k;
}

int main() {
  int arr[] = {1, 2, 2, 3, 4, 5, 5};
  int n = sizeof(arr) / sizeof(arr[0]);
  removeDuplicates(arr, n);
  return 0;
}
```

Output: Array without duplicates: 1 2 3 4 5

29. Find the Common Elements in Two Arrays

- **Problem**: Find common elements between two arrays.

```
#include <stdio.h>
void commonElements(int arr1[], int arr2[], int n1, int n2) {
  for (int i = 0; i < n1; i++) {
    for (int j = 0; j < n2; j++) {
      if (arr1[i] == arr2[j]) {
        printf("%d ", arr1[i]);
      }
    }
  }
}

int main() {
  int arr1[] = {1, 2, 3, 4};
  int arr2[] = {3, 4, 5, 6};
  int n1 = sizeof(arr1) / sizeof(arr1[0]);
  int n2 = sizeof(arr2) / sizeof(arr2[0]);
  commonElements(arr1, arr2, n1, n2);
  return 0;
}
```

Output: 3 4

30. Find the Most Frequent Element in an Array

- **Problem**: Find the most frequent element in an array.

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

int mostFrequent(int arr[], int n) {
  int maxCount = 0, maxElement = arr[0];

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

```
        for (int j = i + 1; j < n; j++) {
            if (arr[i] == arr[j]) {
                count++;
            }
        }
        if (count > maxCount) {
            maxCount = count;
            maxElement = arr[i];
        }
    }
    return maxElement;
}

int main() {
    int arr[] = {3, 1, 4, 1, 2, 1};
    int n = sizeof(arr) / sizeof(arr[0]);
    printf("Most frequent element: %d\n", mostFrequent(arr, n));
    return 0;
}
```

Output: Most frequent element: 1