## Problem 1: Sum of Natural Numbers

**Question**: Write a program that calculates the sum of the first $nn$n natural numbers.

**Code**:

```
c
Copy code
#include <stdio.h>

int main() {
    int n;
    scanf("%d", &n);
    int sum = 0;

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

    printf("%d\n", sum);
    return 0;
}
```

**Explanation**:

- We start by reading the integer $nn$n.
- We initialize a variable `sum` to 0.
- Using a `for` loop, we iterate from 1 to $nn$n, adding each integer to `sum`.
- Finally, we print the total sum.

## Problem 2: Factorial of a Number

**Question**: Write a program that calculates the factorial of a given number $nn$n.

**Code**:

```
c
Copy code
```

```c
#include <stdio.h>

int main() {
    int n;
    scanf("%d", &n);
    long long fact = 1;

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

    printf("%lld\n", fact);
    return 0;
}
```

**Explanation**:

- We read the integer $nn$n.
- We initialize `fact` to 1.
- Using a `for` loop, we multiply `fact` by each integer from 1 to $nn$n.
- The result is printed, which represents $n!n!$n!.

## Problem 3: Print Multiplication Table

**Question**: Write a program that prints the multiplication table for a given number $nn$n up to 10.

**Code**:

```
c
Copy code
#include <stdio.h>

int main() {
    int n;
    scanf("%d", &n);

    for (int i = 1; i <= 10; i++) {
        printf("%d x %d = %d\n", n, i, n * i);
    }
    return 0;
}
```

**Explanation**:

- We read the integer $nn$n.
- Using a `for` loop, we iterate from 1 to 10.
- In each iteration, we calculate and print the product of $nn$n and $ii$i.

## Problem 4: Sum of Even Numbers

**Question**: Write a program that calculates the sum of all even numbers up to $nn$n.

**Code**:

```
c
Copy code
#include <stdio.h>

int main() {
    int n;
    scanf("%d", &n);
    int sum = 0;

    for (int i = 2; i <= n; i += 2) {
        sum += i;
    }

    printf("%d\n", sum);
    return 0;
}
```

**Explanation**:

- We read $nn$n and initialize `sum` to 0.
- The loop starts from 2 and increments by 2 each iteration (only considers even numbers).

- We add each even number to `sum` and print the result.

## Problem 5: Product of Odd Numbers

**Question**: Write a program that calculates the product of all odd numbers up to $nn$n.

**Code**:

```
c
Copy code
#include <stdio.h>

int main() {
    int n;
    scanf("%d", &n);
    long long product = 1;

    for (int i = 1; i <= n; i += 2) {
        product *= i;
    }

    printf("%lld\n", product);
    return 0;
}
```

**Explanation**:

- After reading $nn$n, we initialize `product` to 1.
- The loop starts from 1 and increments by 2 (to consider only odd numbers).
- Each odd number is multiplied to `product`, which is printed at the end.

## Problem 6: Print Numbers in Reverse

**Question**: Write a program that prints numbers from $nn$n down to 1 in reverse order.

**Code**:

```
c
Copy code
```

```
#include <stdio.h>

int main() {
    int n;
    scanf("%d", &n);

    for (int i = n; i >= 1; i--) {
        printf("%d\n", i);
    }
    return 0;
}
```

**Explanation**:

- We read $nn$n.
- The loop starts from $nn$n and decrements until 1.
- Each number is printed in the reverse order.

## Problem 7: Count Divisors

**Question**: Write a program that counts how many numbers between 1 and $nn$n are divisible by a given number $dd$d.

**Code**:

```
c
Copy code
#include <stdio.h>

int main() {
    int n, d;
    scanf("%d %d", &n, &d);
    int count = 0;

    for (int i = 1; i <= n; i++) {
        if (i % d == 0) {
            count++;
        }
    }
    printf("%d\n", count);
    return 0;
}
```

**Explanation**:

- We read $nn$n and $dd$d.
- A `for` loop iterates from 1 to $nn$n.
- We check if each number is divisible by $dd$d using the modulus operator `%`.
- If true, we increment `count` and print the total count at the end.

## Problem 8: Find Power of a Number

**Question**: Write a program that calculates $aba$^$b$ab (a raised to the power of b).

**Code**:

```
c
Copy code
#include <stdio.h>

int main() {
    int a, b;
    scanf("%d %d", &a, &b);
    long long result = 1;

    for (int i = 1; i <= b; i++) {
        result *= a;
    }

    printf("%lld\n", result);
    return 0;
}
```

**Explanation**:

- We read $aa$a and $bb$b.

- We initialize `result` to 1.
- A loop multiplies `result` by $a$a, $b$b times.
- The final result is printed.

## Problem 9: Print All Factors

**Question**: Write a program that prints all factors of a given number $n$n.

**Code**:

```
c
Copy code
#include <stdio.h>

int main() {
    int n;
    scanf("%d", &n);

    for (int i = 1; i <= n; i++) {
        if (n % i == 0) {
            printf("%d\n", i);
        }
    }
    return 0;
}
```

**Explanation**:

- After reading $n$n, we use a loop from 1 to $n$n.
- We check if $i$i is a divisor of $n$n using `%`.
- If it is, we print it as a factor.

## Problem 10: Sum of Digits

**Question**: Write a program that calculates the sum of the digits of a given number $n$n.

**Code**:

```
c
```

Copy code

```
#include <stdio.h>

int main() {
    int n;
    scanf("%d", &n);
    int sum = 0;

    while (n > 0) {
        sum += n % 10; // Add last digit to sum
        n /= 10; // Remove last digit
    }

    printf("%d\n", sum);
    return 0;
}
```

**Explanation**:

- We read $nn$n and initialize `sum` to 0.
- A `while` loop runs as long as $nn$n is greater than 0.
- The last digit is added to `sum`, and $nn$n is divided by 10 to remove that last digit.
- Finally, we print the sum of the digits.

## Problem 11: Print Fibonacci Sequence

**Question**: Write a program that prints the Fibonacci sequence up to the $nn$n-th term.

**Code**:

c

Copy code

```
#include <stdio.h>

int main() {
    int n;
    scanf("%d", &n);
    int a = 0, b = 1;
    for (int i = 1; i <= n; i++) {
        printf("%d\n", a);
        int next = a + b;
        a = b;
        b = next;
    }
    return 0;
}
```

**Explanation**:

- We read $nn$n.
- We initialize the first two Fibonacci numbers $aa$a and $bb$b to 0 and 1, respectively.
- A loop iterates $nn$n times, printing $aa$a each time.
- The next Fibonacci number is calculated by adding $aa$a and $bb$b, and we update $aa$a and $bb$b for the next iteration.

## Problem 12: Count Prime Numbers

**Question**: Write a program that counts how many prime numbers exist between 1 and $nn$n.

**Code**:

```
c
Copy code
#include <stdio.h>

int main() {
    int n;
    scanf("%d", &n);
    int count = 0;

    for (int i = 2; i <= n; i++) {
        int isPrime = 1; // Assume it's prime
        for (int j = 2; j * j <= i; j++) {
            if (i % j == 0) {
                isPrime = 0; // Not prime
                break;
            }
        }
        count += isPrime; // Increment if prime
    }

    printf("%d\n", count);
    return 0;
}
```

**Explanation**:

- We read $n$n and initialize count to 0.
- A loop iterates from 2 to $n$n.
- For each number, we assume it's prime and check for factors using a nested loop.
- If we find a divisor, we set `isPrime` to 0.
- At the end, we increment `count` if the number is prime.

## Problem 13: Print Prime Numbers

**Question**: Write a program that prints all prime numbers between 1 and $n$n.

**Code**:

```
c
Copy code
#include <stdio.h>

int main() {
    int n;
    scanf("%d", &n);

    for (int i = 2; i <= n; i++) {
        int isPrime = 1; // Assume it's prime
        for (int j = 2; j * j <= i; j++) {
            if (i % j == 0) {
                isPrime = 0; // Not prime
                break;
            }
        }
        if (isPrime) {
            printf("%d\n", i); // Print if prime
        }
    }
    return 0;
}
```

**Explanation**:

- This code is similar to Problem 12.
- Instead of counting primes, we print each prime found during the iteration.

## Problem 14: Greatest Common Divisor (GCD)

**Question**: Write a program that calculates the greatest common divisor (GCD) of two numbers *aa*a and *bb*b.

**Code**:

```
c
Copy code
#include <stdio.h>

int main() {
    int a, b;
    scanf("%d %d", &a, &b);

    while (b != 0) {
        int temp = b;
        b = a % b; // Calculate remainder
        a = temp; // Update a
    }

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

**Explanation**:

- We read two integers *aa*a and *bb*b.
- Using the Euclidean algorithm, we continuously update *aa*a and *bb*b until *bb*b becomes 0.
- When *bb*b is 0, *aa*a holds the GCD, which we print.

## Problem 15: Least Common Multiple (LCM)

**Question**: Write a program that calculates the least common multiple (LCM) of two numbers *aa*a and *bb*b.

**Code**:

```
c
Copy code
#include <stdio.h>

int main() {
    int a, b;
    scanf("%d %d", &a, &b);
    int gcd;

    // Find GCD using the Euclidean algorithm
    int tempA = a, tempB = b;
    while (tempB != 0) {
        int temp = tempB;
        tempB = tempA % tempB;
        tempA = temp;
    }
    gcd = tempA;

    // Calculate LCM
    long long lcm = (a / gcd) * b; // (a * b) / gcd is also valid

    printf("%lld\n", lcm);
    return 0;
}
```

**Explanation**:

- We first find the GCD of $aa$a and $bb$b using the Euclidean algorithm.
- Then, we calculate the LCM using the formula *LCM=a×bGCD\text{LCM} = \frac{a \times b}{\text{GCD}}*LCM=GCDa×b (though the formula *aGCD×b\frac{a}{\text{GCD}} \times b*GCDa ×b helps avoid overflow).
- Finally, we print the LCM.

## Problem 16: Sum of Squares

**Question**: Write a program that calculates the sum of the squares of the first $nn$n natural numbers.

**Code**:

```
c
Copy code
#include <stdio.h>

int main() {
    int n;
    scanf("%d", &n);
    int sum = 0;

    for (int i = 1; i <= n; i++) {
        sum += i * i; // Add square of i
    }

    printf("%d\n", sum);
    return 0;
}
```

**Explanation**:

- We read $nn$n and initialize `sum` to 0.
- Using a loop, we calculate the square of each number from 1 to $nn$n and add it to `sum`.
- Finally, we print the result.

## Problem 17: Check Armstrong Number

**Question**: Write a program that checks if a given number $nn$n is an Armstrong number.

**Code**:

```
c
Copy code
#include <stdio.h>
#include <math.h>

int main() {
    int n;
    scanf("%d", &n);
    int original = n, sum = 0, digits = 0;

    // Count digits
    while (original > 0) {
        original /= 10;
        digits++;
    }

    original = n; // Reset original
    while (original > 0) {
        int lastDigit = original % 10;
        sum += pow(lastDigit, digits); // Raise to the power of number
of digits
        original /= 10; // Remove last digit
    }

    if (sum == n) {
        printf("Yes\n");
    } else {
        printf("No\n");
    }
    return 0;
}
```

**Explanation**:

- We read $nn$n and store it in `original`.
- We count the number of digits in $nn$n.
- We calculate the sum of each digit raised to the power of the total number of digits.
- Finally, we check if this sum equals $nn$n and print "Yes" or "No".

## Problem 18: Reverse a Number

**Question**: Write a program that reverses the digits of a given number $nn$n.

**Code**:

```
c
Copy code
#include <stdio.h>

int main() {
    int n;
    scanf("%d", &n);
    int reversed = 0;

    while (n > 0) {
        int lastDigit = n % 10;
        reversed = reversed * 10 + lastDigit; // Shift and add last
digit
        n /= 10; // Remove last digit
    }

    printf("%d\n", reversed);
    return 0;
}
```

**Explanation**:

- We read $nn$n and initialize `reversed` to 0.
- Using a `while` loop, we extract the last digit and shift `reversed` to the left (by multiplying by 10) to add the last digit.
- Finally, we print the reversed number.

## Problem 19: Sum of Cubes

**Question**: Write a program that calculates the sum of the cubes of the first 𝑛𝑛n natural numbers.

**Code**:

```
c
Copy code
#include <stdio.h>

int main() {
    int n;
    scanf("%d", &n);
    int sum = 0;

    for (int i = 1; i <= n; i++) {
        sum += i * i * i; // Add cube of i
    }

    printf("%d\n", sum);
    return 0;
}
```

**Explanation**:

- We read 𝑛𝑛n and initialize `sum` to 0.
- Using a loop, we calculate the cube of each number from 1 to 𝑛𝑛n and add it to `sum`.
- Finally, we print the result.

## Problem 20: Find Minimum and Maximum

**Question**: Write a program that finds the minimum and maximum of a sequence of 𝑛𝑛n numbers.

**Code**:

```
c
Copy code
```

```c
#include <stdio.h>

int main() {
    int n;
    scanf("%d", &n);
    int min, max;

    for (int i = 1; i <= n; i++) {
        int number;
        scanf("%d", &number);

        if (i == 1) {
            min = max = number; // Initialize min and max
        } else {
            if (number < min) min = number; // Update min
            if (number > max) max = number; // Update max
        }
    }

    printf("%d %d\n", min, max);
    return 0;
}
```

**Explanation**:

- We read $nn$n.
- The loop reads $nn$n integers and checks the first number to set both `min` and `max`.
- For each subsequent number, we update `min` and `max` as needed.
- Finally, we print both values.

## Problem 21: Sum of Even Digits

**Question**: Write a program that calculates the sum of even digits of a given number $nn$n.

**Code**:

```
c
Copy code
```

```
#include <stdio.h>

int main() {
    int n;
    scanf("%d", &n);
    int sum = 0;

    while (n > 0) {
        int lastDigit = n % 10;
        if (lastDigit % 2 == 0) { // Check if even
            sum += lastDigit;
        }
        n /= 10; // Remove last digit
    }

    printf("%d\n", sum);
    return 0;
}
```

**Explanation**:

- We read $n$n and initialize `sum` to 0.
- Using a `while` loop, we extract each digit.
- If a digit is even, we add it to `sum`.
- Finally, we print the sum of even digits.

## Problem 22: Count Vowels

**Question**: Write a program that counts the number of vowels in a given string.

**Code**:

```
c
Copy code
#include <stdio.h>

int main() {
    char str[100]; // Array to hold input string
    fgets(str, sizeof(str), stdin);
    int count = 0;

    for (int i = 0; str[i] != '\0'; i++) {
        char c = str[i];
        if (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u'
||
            c == 'A' || c == 'E' || c == 'I' || c == 'O' || c == 'U')
{
            count++; // Increment if vowel
        }
    }

    printf("%d\n", count);
    return 0;
}
```

**Explanation**:

- We declare a character array `str` to hold the input string.
- Using `fgets`, we read the string.
- We iterate through each character in the string, checking if it is a vowel.
- If it is, we increment `count` and print the total.

## Problem 23: Check Palindrome

**Question**: Write a program that checks if a given string is a palindrome.

**Code**:

```
c
Copy code
#include <stdio.h>
#include <string.h>

int main() {
    char str[100];
    fgets(str, sizeof(str), stdin);
    int len = strlen(str) - 1; // Exclude newline character
    int isPalindrome = 1; // Assume true
    for (int i = 0; i < len / 2; i++) {
        if (str[i] != str[len - 1 - i]) {
            isPalindrome = 0; // Not a palindrome
            break;
        }
    }

    if (isPalindrome) {
        printf("Yes\n");
    } else {
        printf("No\n");
    }

    return 0;
}
```

**Explanation**:

- We read the string and calculate its length, excluding the newline character.
- A loop checks characters from the start and end of the string moving toward the center.
- If any characters don't match, we set `isPalindrome` to 0.
- Finally, we print "Yes" or "No".

## Problem 24: Count Consonants

**Question**: Write a program that counts the number of consonants in a given string.

**Code**:

```
c
Copy code
#include <stdio.h>

int main() {
    char str[100];
    fgets(str, sizeof(str), stdin);
    int count = 0;
    for (int i = 0; str[i] != '\0'; i++) {
        char c = str[i];
        if ((c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z')) { //
Check if alphabet
            if (!(c == 'a' || c == 'e' || c == 'i' || c == 'o' || c ==
'u' ||
                  c == 'A' || c == 'E' || c == 'I' || c == 'O' || c ==
'U')) {
                count++; // Increment if consonant
            }
        }
    }

    printf("%d\n", count);
    return 0;
}
```

**Explanation**:

- We read the string and initialize `count` to 0.
- The loop iterates through each character, checking if it is an alphabet.
- If it's not a vowel, we increment `count`.
- Finally, we print the total count of consonants.

## Problem 25: Calculate Average

**Question**: Write a program that calculates the average of a series of numbers.

**Code**:

```
c
Copy code
#include <stdio.h>

int main() {
    int n;
    scanf("%d", &n);
    double sum = 0;
    for (int i = 1; i <= n; i++) {
        double number;
        scanf("%lf", &number);
        sum += number; // Add each number to sum
    }

    double average = sum / n; // Calculate average
    printf("%lf\n", average);
    return 0;
}
```

**Explanation**:

- We read $nn$n and initialize `sum` to 0.
- A loop reads $nn$n numbers and adds them to `sum`.
- Finally, we calculate and print the average.

## Problem 26: Count Digits

**Question**: Write a program that counts the number of digits in a given number $nn$n.

**Code**:

```
c
Copy code
#include <stdio.h>

int main() {
    int n;
    scanf("%d", &n);
    int count = 0;

    while (n > 0) {
        n /= 10; // Remove last digit
        count++; // Increment count
    }

    printf("%d\n", count);
    return 0;
}
```

**Explanation**:

- We read $nn$n and initialize count to 0.
- Using a `while` loop, we continuously divide $nn$n by 10 to remove the last digit and increment `count`.
- Finally, we print the total count of digits.

## Problem 27: Count Words

**Question**: Write a program that counts the number of words in a given string.

**Code**:

c

Copy code

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

int main() {
    char str[100];
    fgets(str, sizeof(str), stdin);
    int count = 0;
    int inWord = 0;

    for (int i = 0; str[i] != '\0'; i++) {
        if (str[i] == ' ' || str[i] == '\n') {
            inWord = 0; // End of a word
        } else {
            if (inWord == 0) {
                count++; // New word
                inWord = 1; // Inside a word
            }
        }
    }

    printf("%d\n", count);
    return 0;
}
```

**Explanation**:

- We read the string and initialize `count` and `inWord`.
- The loop iterates through each character.
- If a space or newline is found, we set `inWord` to 0, indicating the end of a word.
- When we encounter a non-space character after being outside a word, we increment `count`.

## Problem 28: Find Largest Digit

**Question**: Write a program that finds the largest digit in a given number $n$n.

**Code**:

```
c
Copy code
#include <stdio.h>

int main() {
    int n;
    scanf("%d", &n);
    int largest = 0;

    while (n > 0) {
        int lastDigit = n % 10; // Get last digit
        if (lastDigit > largest) {
            largest = lastDigit; // Update largest
        }
        n /= 10; // Remove last digit
    }

    printf("%d\n", largest);
    return 0;
}
```

**Explanation**:

- We read $nn$n and initialize `largest` to 0.
- A `while` loop extracts the last digit, comparing it with `largest`.
- If it's greater, we update `largest`.
- Finally, we print the largest digit found.

## Problem 29: Count Occurrences

**Question**: Write a program that counts how many times a digit appears in a given number $nn$n.

**Code**:

```
c
Copy code
#include <stdio.h>

int main() {
    int n, digit;
    scanf("%d %d", &n, &digit);
    int count = 0;

    while (n > 0) {
        int lastDigit = n % 10;
        if (lastDigit == digit) {
            count++; // Increment if found
        }
        n /= 10; // Remove last digit
    }

    printf("%d\n", count);
    return 0;
}
```

**Explanation**:

- We read $nn$n and the target `digit` to search.
- A `while` loop checks each digit of $nn$n.

- If a match is found, we increment `count`.
- Finally, we print the number of occurrences.

## Problem 30: Fibonacci Number

**Question**: Write a program that checks if a number *nn*n is a Fibonacci number.

**Code**:

```
c
Copy code
#include <stdio.h>

int main() {
    int n;
    scanf("%d", &n);
    int a = 0, b = 1, fib = 0;

    while (fib < n) {
        fib = a + b; // Calculate next Fibonacci number
        a = b;
        b = fib;
    }

    if (fib == n) {
        printf("Yes\n");
    } else {
        printf("No\n");
    }

    return 0;
}
```

**Explanation**:

- We read *nn*n.
- Starting with the first two Fibonacci numbers (0 and 1), we calculate the next Fibonacci number until it reaches or exceeds *nn*n.
- Finally, we check if *fibfib*fib equals *nn*n and print the result.

## Problem 31: Find Prime Factors

**Question**: Write a program that finds all prime factors of a given number $nn$n.

**Code**:

```
c
Copy code
#include <stdio.h>

int main() {
    int n;
    scanf("%d", &n);

    for (int i = 2; i <= n; i++) {
        while (n % i == 0) {
            printf("%d\n", i); // Print prime factor
            n /= i; // Divide out the prime factor
        }
    }
    return 0;
}
```

**Explanation**:

- We read $nn$n.
- A loop checks each number from 2 up to $nn$n.
- If $ii$i is a factor of $nn$n, we print it and divide $nn$n by $ii$i until it's no longer a factor.
- This process continues until all prime factors are found.

## Problem 32: Generate Pascal's Triangle

**Question**: Write a program to generate the first $nn$n rows of Pascal's triangle.

**Code**:

```
c
Copy code
```

```
#include <stdio.h>

int main() {
    int n;
    scanf("%d", &n);

    for (int line = 0; line < n; line++) {
        int C = 1; // Initialize coefficient
        for (int i = 0; i <= line; i++) {
            printf("%d ", C); // Print coefficient
            C = C * (line - i) / (i + 1); // Calculate next
coefficient
        }
        printf("\n");
    }
    return 0;
}
```

**Explanation**:

- We read $nn$n.
- A nested loop generates coefficients using the properties of binomial coefficients.
- The inner loop calculates each coefficient using the previous one, printing them in a line.

## Problem 33: Merge Two Sorted Arrays

**Question**: Write a program to merge two sorted arrays into a single sorted array.

**Code**:

```
c
Copy code
#include <stdio.h>

int main() {
    int n1, n2;
    scanf("%d", &n1);
    int arr1[n1];
    for (int i = 0; i < n1; i++) {
        scanf("%d", &arr1[i]); // Read first sorted array
    }

    scanf("%d", &n2);
    int arr2[n2];
    for (int i = 0; i < n2; i++) {
        scanf("%d", &arr2[i]); // Read second sorted array
    }

    int merged[n1 + n2], i = 0, j = 0, k = 0;

    while (i < n1 && j < n2) {
        if (arr1[i] < arr2[j]) {
            merged[k++] = arr1[i++]; // Add from arr1
        } else {
            merged[k++] = arr2[j++]; // Add from arr2
        }
    }

    while (i < n1) {
        merged[k++] = arr1[i++]; // Add remaining from arr1
    }

    while (j < n2) {
        merged[k++] = arr2[j++]; // Add remaining from arr2
    }

    for (int i = 0; i < n1 + n2; i++) {
        printf("%d ", merged[i]); // Print merged array
    }
    printf("\n");
    return 0;
}
```

**Explanation**:

- We read sizes and elements of two sorted arrays.

- Using a while loop, we compare and merge elements from both arrays into a new array.
- After merging, we print the merged array.

## Problem 34: Find Common Elements

**Question**: Write a program that finds common elements in two arrays.

**Code**:

```
c
Copy code
#include <stdio.h>

int main() {
    int n1, n2;
    scanf("%d", &n1);
    int arr1[n1];
    for (int i = 0; i < n1; i++) {
        scanf("%d", &arr1[i]); // Read first array
    }

    scanf("%d", &n2);
    int arr2[n2];
    for (int i = 0; i < n2; i++) {
        scanf("%d", &arr2[i]); // Read second array
    }

    for (int i = 0; i < n1; i++) {
        for (int j = 0; j < n2; j++) {
            if (arr1[i] == arr2[j]) {
                printf("%d\n", arr1[i]); // Print common element
                break; // Avoid duplicates
            }
        }
    }

    return 0;
}
```

**Explanation**:

- We read the sizes and elements of two arrays.
- A nested loop compares each element of the first array with each element of the second array.
- If a common element is found, it is printed.

## Problem 35: Count Occurrences of Each Element

**Question**: Write a program that counts occurrences of each element in an array.

**Code**:

```
c
Copy code
#include <stdio.h>

int main() {
    int n;
    scanf("%d", &n);
    int arr[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]); // Read array
    }

    for (int i = 0; i < n; i++) {
        int count = 1;
        if (arr[i] == -1) continue; // Skip counted elements

        for (int j = i + 1; j < n; j++) {
            if (arr[i] == arr[j]) {
                count++; // Count occurrences
                arr[j] = -1; // Mark as counted
            }
        }
        printf("%d occurs %
```

4o mini

**You said:**

continue

***ChatGPT said:***

ChatGPT

c

Copy code

```
        printf("%d occurs %d times\n", arr[i], count); // Print count of each element
    }

    return 0;
}
```

**Explanation**:

- We read the size *nn*n and the elements of the array.
- We iterate through each element of the array with an outer loop.
- For each element, we check for duplicates with an inner loop.
- If a duplicate is found, we increment the `count` and mark the duplicate element as counted by setting it to `-1`.
- Finally, we print how many times each unique element occurs in the array.

### Problem 36: Reverse a String

**Question**: Write a program that reverses a given string.

**Code**:

c

Copy code

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

int main() {
    char str[100];
    fgets(str, sizeof(str), stdin); // Read the string
    int len = strlen(str) - 1; // Exclude newline character

    for (int i = len; i >= 0; i--) {
        putchar(str[i]); // Print characters in reverse order
    }
    printf("\n");
    return 0;
}
```

**Explanation**:

- We read a string into the character array `str` using `fgets`.
- We determine the length of the string and exclude the newline character.
- A loop iterates from the end of the string to the beginning, printing each character using `putchar`.
- Finally, we print a newline for better formatting.

### Problem 37: Remove Vowels from a String

**Question**: Write a program that removes all vowels from a given string.

**Code**:

```
c
Copy code
#include <stdio.h>
#include <string.h>

int main() {
    char str[100], result[100];
    fgets(str, sizeof(str), stdin); // Read the string
    int index = 0;

    for (int i = 0; str[i] != '\0'; i++) {
        char c = str[i];
        if (!(c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u'
||
              c == 'A' || c == 'E' || c == 'I' || c == 'O' || c ==
'U')) {
            result[index++] = c; // Add non-vowel to result
        }
    }
    result[index] = '\0'; // Null-terminate the result string
    printf("%s\n", result); // Print the string without vowels
    return 0;
}
```

**Explanation**:

- We read a string into `str`.
- We iterate through each character in `str`, checking if it's a vowel.
- If it's not a vowel, we add it to the `result` array and increment `index`.
- Finally, we null-terminate the `result` string and print it.

## Problem 38: Check Armstrong Number

**Question**: Write a program that checks if a number *nn*n is an Armstrong number.

**Code**:

```
c
Copy code
#include <stdio.h>
#include <math.h>

int main() {
    int n, original, remainder, digits = 0, sum = 0;
    scanf("%d", &n);
    original = n;

    // Count the number of digits
    while (original != 0) {
        original /= 10;
        digits++;
    }
    original = n;

    // Calculate the sum of the cubes of each digit
    while (original != 0) {
        remainder = original % 10;
        sum += pow(remainder, digits);
        original /= 10;
    }

    // Check if the number is an Armstrong number
    if (sum == n) {
        printf("Yes\n");
    } else {
        printf("No\n");
    }

    return 0;
}
```

**Explanation**:

- We read a number $nn$n.
- We calculate the number of digits in $nn$n using a loop.
- In a second loop, we compute the sum of each digit raised to the power of the number of digits.
- Finally, we check if the sum equals $nn$n and print "Yes" or "No".

## Problem 39: Calculate Factorial

**Question**: Write a program to calculate the factorial of a number $nn$n.

**Code**:

```
c
Copy code
#include <stdio.h>

int main() {
    int n;
    scanf("%d", &n);
    long long factorial = 1; // Use long long to handle large results

    for (int i = 1; i <= n; i++) {
        factorial *= i; // Multiply each number to get factorial
    }

    printf("%lld\n", factorial); // Print the factorial
    return 0;
}
```

**Explanation**:

- We read a number $nn$n.
- We initialize `factorial` to 1.
- A loop iterates from 1 to $nn$n, multiplying `factorial` by each integer.
- Finally, we print the resulting factorial.

## Problem 40: Count Unique Characters

**Question**: Write a program that counts the number of unique characters in a string.

**Code**:

```
c
Copy code
#include <stdio.h>
#include <string.h>

int main() {
    char str[100];
    fgets(str, sizeof(str), stdin); // Read the string
    int count[256] = {0}; // Array to track character occurrences
    int uniqueCount = 0;

    for (int i = 0; str[i] != '\0'; i++) {
        count[(unsigned char)str[i]]++; // Increment count for each
character
    }
    // Count unique characters
    for (int i = 0; i < 256; i++) {
        if (count[i] == 1) {
            uniqueCount++; // Increment unique count
        }
    }

    printf("%d\n", uniqueCount); // Print the number of unique
characters
    return 0;
}
```

**Explanation**:

- We read a string into `str`.
- We declare an array `count` of size 256 to track occurrences of each character.
- We iterate through the string, incrementing the count for each character.
- After that, we count characters that appear exactly once, updating `uniqueCount`.
- Finally, we print the count of unique characters.