



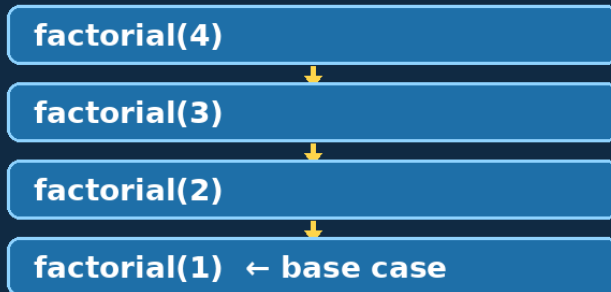
SOLVED C RECURSION PROBLEMS

Step-by-Step Programs for Students & GATE Aspirants

💡 Quick Reading Guide

This document contains the problem statements and C solutions for seven recursion-related practice problems. Focus on the base condition, recursive call, and what happens when the function returns.

RECURSION → CALL STACK → RETURN



Calls go down ↓

Base case stops recursion

Then values return ↑

LIFO • Function Call Stack

📖 **Previous lesson:** If you are new to recursion, first study the basic **1 + 2 + 3 + 4 + 5 recursion example** and then use this document for practice.

💎 Question 1: Factorial Using Recursion

Problem: Find the factorial of a given number using recursion.

🔗 Recursion Idea

For $n > 1$, $\text{factorial}(n) = n \times \text{factorial}(n - 1)$. The recursion stops at $n = 0$ or $n = 1$.

Solution:

```
#include <stdio.h>

long long factorial(int n)
{
    if (n == 0 || n == 1)
        return 1;

    return n * factorial(n - 1);
}

int main()
{
    int n;

    printf("Enter a number: ");
    scanf("%d", &n);
```

```

printf("Factorial = %lld", factorial(n));

return 0;
}

```

✓ **Example:** $5! = 120$

◆ Question 2: Sum of Digits — Without and With Recursion

Problem: A 5-digit positive integer is entered through the keyboard. Calculate the sum of its digits without recursion and using recursion.

🔗 Recursion Idea

Use $n \% 10$ to obtain the last digit and $n / 10$ to remove it. The recursive version repeats this process until n becomes 0.

Solution:

```

/* Without recursion */
#include <stdio.h>

int main()
{
    int n, sum = 0;

    printf("Enter a 5-digit number: ");
    scanf("%d", &n);

    while (n != 0)
    {
        sum = sum + n % 10;
        n = n / 10;
    }

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

    return 0;
}

/* Using recursion */
#include <stdio.h>

int sumDigits(int n)
{
    if (n == 0)
        return 0;

    return (n % 10) + sumDigits(n / 10);
}

int main()
{
    int n;

    printf("Enter a 5-digit number: ");
    scanf("%d", &n);

    printf("Sum of digits = %d", sumDigits(n));
}

```

```
    return 0;
}
```

✂ The two versions above are separate programs. Compile and run one version at a time.

✓ **Example:** $12345 \rightarrow 1 + 2 + 3 + 4 + 5 = 15$

◆ Question 3: Prime Factors Using Recursion

Problem: A positive integer is entered through the keyboard. Find its prime factors and modify the function to obtain the prime factors recursively.

🔗 Recursion Idea

Start with divisor 2. If it divides n , print it and recursively continue with $n / \text{divisor}$. Otherwise, try the next divisor. Stop when n becomes 1.

Solution:

```
#include <stdio.h>

void primeFactors(int n, int divisor)
{
    if (n == 1)
        return;

    if (n % divisor == 0)
    {
        printf("%d ", divisor);
        primeFactors(n / divisor, divisor);
    }
    else
    {
        primeFactors(n, divisor + 1);
    }
}

int main()
{
    int n;

    printf("Enter a positive integer: ");
    scanf("%d", &n);

    printf("Prime factors: ");
    primeFactors(n, 2);

    return 0;
}
```

✓ **Example:** $60 \rightarrow 2 \times 2 \times 3 \times 5$

◆ Question 4: First 25 Fibonacci Numbers

Problem: Write a recursive function to obtain the first 25 numbers of a Fibonacci sequence. The sequence begins 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89...

🔗 Recursion Idea

Each term is the sum of the two preceding terms: $F(n) = F(n - 1) + F(n - 2)$. The first two terms are 1 and 1.

Solution:

```
#include <stdio.h>

int fibonacci(int n)
{
    if (n == 1 || n == 2)
        return 1;

    return fibonacci(n - 1) + fibonacci(n - 2);
}

int main()
{
    int i;

    printf("First 25 Fibonacci numbers:\n");

    for (i = 1; i <= 25; i++)
        printf("%d ", fibonacci(i));

    return 0;
}
```

✓ **Example:** The sequence starts: 1, 1, 2, 3, 5, 8, 13...

Question 5: Decimal to Binary Using Recursion

Problem: A positive integer is entered through the keyboard. Find its binary equivalent using recursion.

Recursion Idea

Divide the number by 2 recursively. The remainder $n \% 2$ gives each binary digit. Printing after the recursive call produces the digits in the correct order.

Solution:

```
#include <stdio.h>

void binary(int n)
{
    if (n > 1)
        binary(n / 2);

    printf("%d", n % 2);
}

int main()
{
    int n;

    printf("Enter a positive integer: ");
    scanf("%d", &n);

    printf("Binary equivalent = ");
    binary(n);

    return 0;
}
```

✓ **Example:** 10 → 1010

◆ Question 6: Display the Binary Equivalent

Problem: Write a function to find the binary equivalent of a given decimal integer and display it.

🔗 Recursion Idea

This problem uses the same core recursive technique as Question 5: recursively divide by 2 and display the remainder while the recursive calls return.

Solution:

```
#include <stdio.h>

void displayBinary(int n)
{
    if (n > 1)
        displayBinary(n / 2);

    printf("%d", n % 2);
}

int main()
{
    int n;

    printf("Enter a decimal integer: ");
    scanf("%d", &n);

    printf("Binary equivalent = ");
    displayBinary(n);

    return 0;
}
```

✓ **Example:** 25 → 11001

◆ Question 7: Factorial of an Integer

Problem: Write a function to calculate the factorial value of any integer entered through the keyboard.

🔗 Recursion Idea

The recursive pattern is $\text{factorial}(n) = n \times \text{factorial}(n - 1)$. The base condition is $\text{factorial}(0) = 1$.

Solution:

```
#include <stdio.h>

long long fact(int n)
{
    if (n == 0)
        return 1;

    return n * fact(n - 1);
}

int main()
{
    int n;

    printf("Enter an integer: ");
```

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

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

return 0;
}
```

✓ **Example:** $6! = 720$

Important Points to Remember

- A recursive function must have a base condition.
- The recursive call should move the problem toward the base condition.
- Recursive calls are stored in the function call stack.
- A function may contain one recursive call or more than one recursive call.
- Tracing the calls and return values on paper is one of the best ways to understand recursion.



The Creators of C Language

Dennis Ritchie
Creator of C Language
“C is a powerful, flexible language that has stood the test of time.”

Ken Thompson
Co-creator of Unix & C
“The best programmers write simple and clear code.”

Brian Kernighan
Co-author of The C Programming Language
“Simplicity, clarity and good design are timeless.”

```
#include <stdio.h>
int main()
{
    printf("Hello C!\n");
    return 0;
}
```

THE C PROGRAMMING LANGUAGE

EngineersTutor.com
Your Learning Partner for Programming and GATE Preparation

EngineersTutor
Subscribe for More Tutorials

Learn Practice Grow

 **KEEP LEARNING • KEEP PROGRAMMING** 

Master C Programming One Problem at a Time

 C • RECURSION • PROGRAMMING • GATE 

 **EngineersTutor.com**

► **EngineersTutor — YouTube**

Learn • Practice • Understand • Succeed