AMRITA SCHOOL OF COMPUTING

DESIGN AND ANALYSIS OF ALGORITHMS (23CSE211)

Name: P.Jeeven Sandeep

Roll No.: CH.SC.U4CSE24134

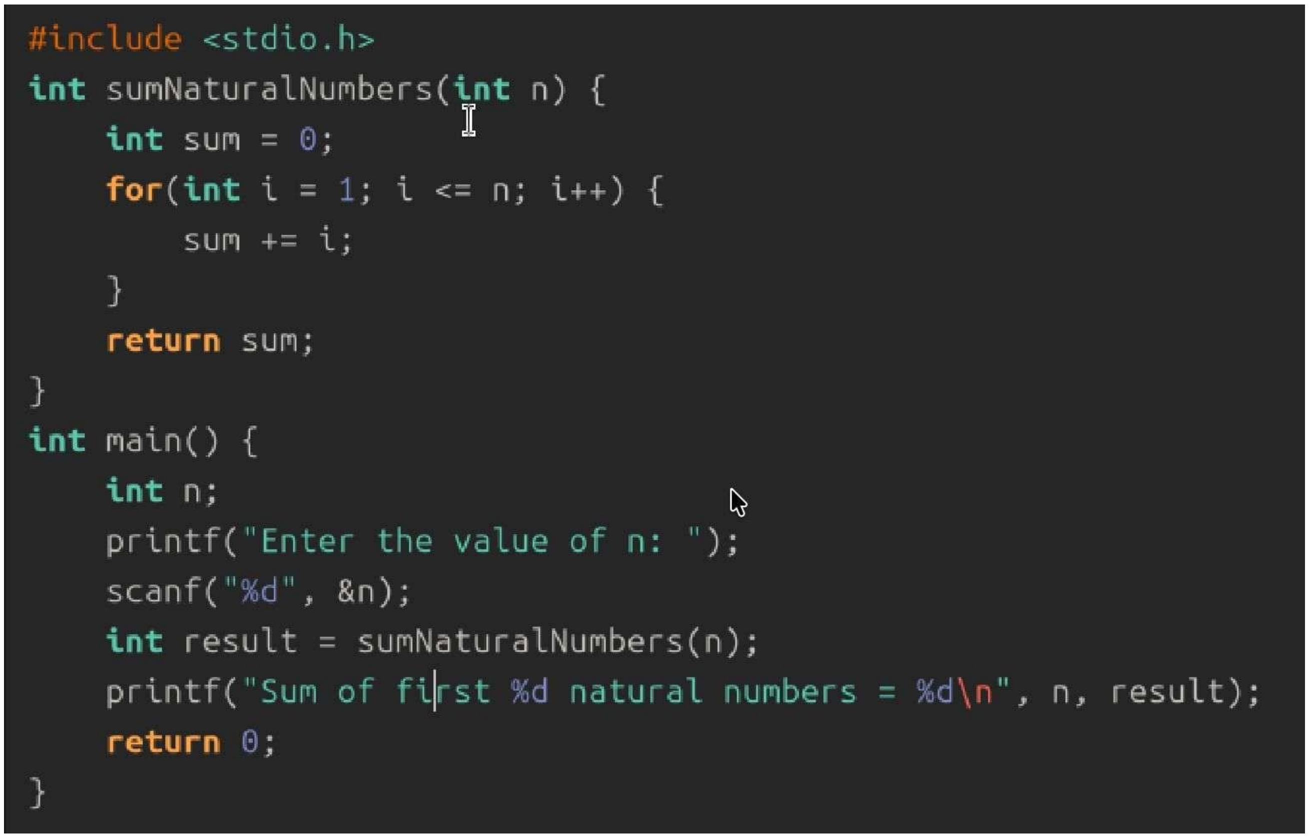
Class: BTech (CSE-B)

School: Amrita School of Computing,
Chennai Campus.

LAB-1

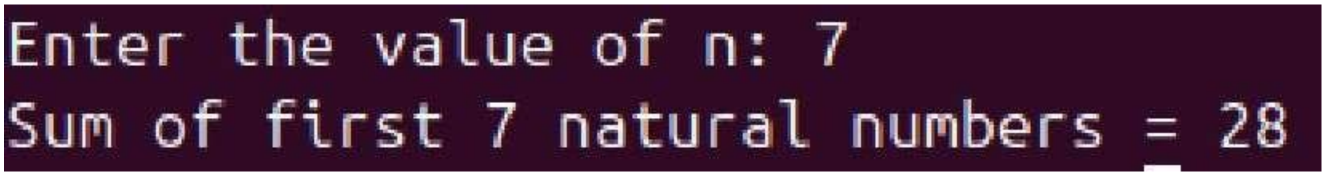
1) Write a program to find sum of n natural numbers (using user defined function)

Code:



```c
#include <stdio.h>
int sumNaturalNumbers(int n) {
    int sum = 0;
    for(int i = 1; i <= n; i++) {
        sum += i;
    }
    return sum;
}
int main() {
    int n;
    printf("Enter the value of n: ");
    scanf("%d", &n);
    int result = sumNaturalNumbers(n);
    printf("Sum of first %d natural numbers = %d\n", n, result);
    return 0;
}
```

Output:



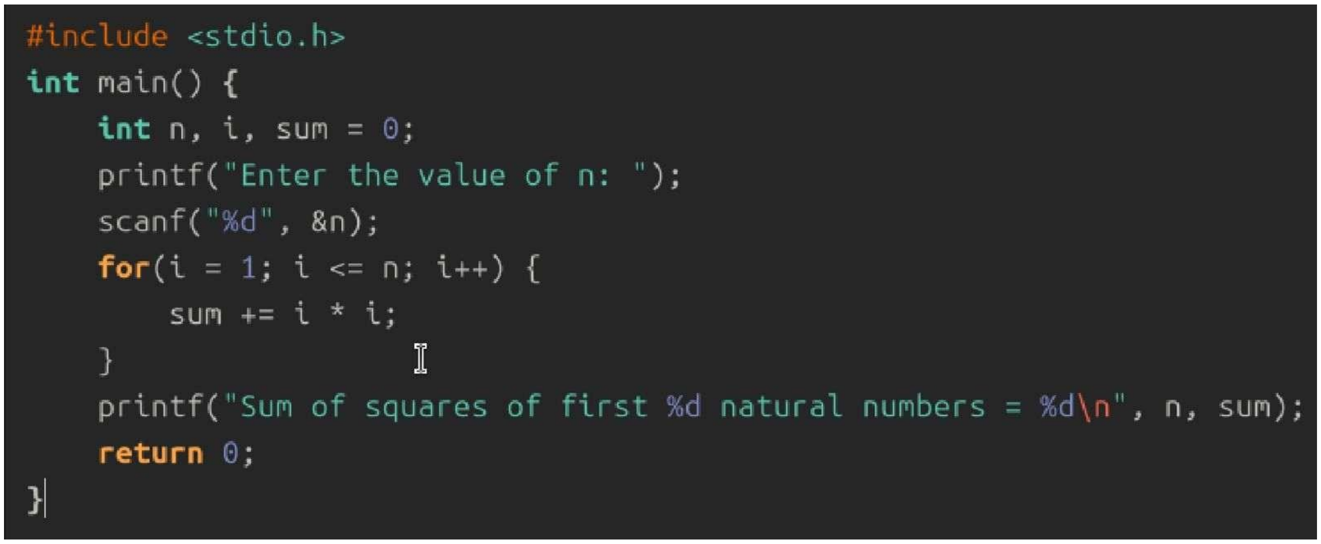
```
Enter the value of n: 7
Sum of first 7 natural numbers = 28
```

Space Complexity:

The space complexity of this program is **O(1)** because it uses only a few fixed variables. The amount of memory needed does not increase with the input size, so the space remains constant.

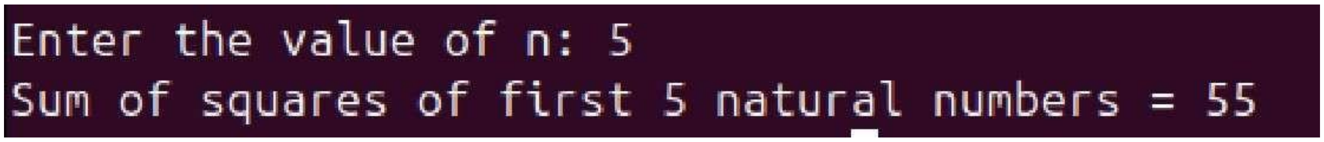
2) Write a program to find sum of squares of first n natural numbers

Code:



```c
#include <stdio.h>
int main() {
    int n, i, sum = 0;
    printf("Enter the value of n: ");
    scanf("%d", &n);
    for(i = 1; i <= n; i++) {
        sum += i * i;
    }
    printf("Sum of squares of first %d natural numbers = %d\n", n, sum);
    return 0;
}
```

Output:



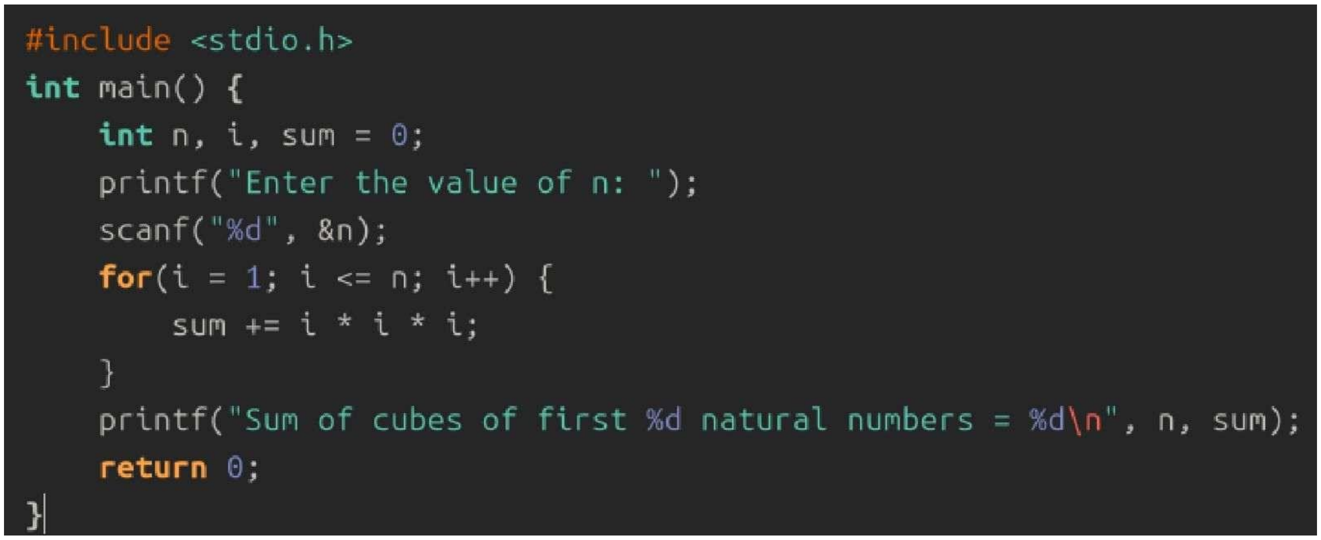
```
Enter the value of n: 5
Sum of squares of first 5 natural numbers = 55
```

Space Complexity:

The space complexity of this program is **O(1)** because it does not use recursion, arrays, or dynamic memory. It only uses a fixed set of variables, so the memory usage does not increase with *n*.

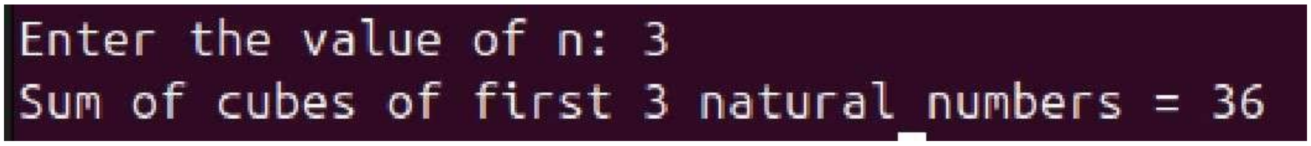
3) Write a program to find sum of cubes of first n natural numbers

Code:



```c
#include <stdio.h>
int main() {
    int n, i, sum = 0;
    printf("Enter the value of n: ");
    scanf("%d", &n);
    for(i = 1; i <= n; i++) {
        sum += i * i * i;
    }
    printf("Sum of cubes of first %d natural numbers = %d\n", n, sum);
    return 0;
}
```

Output:



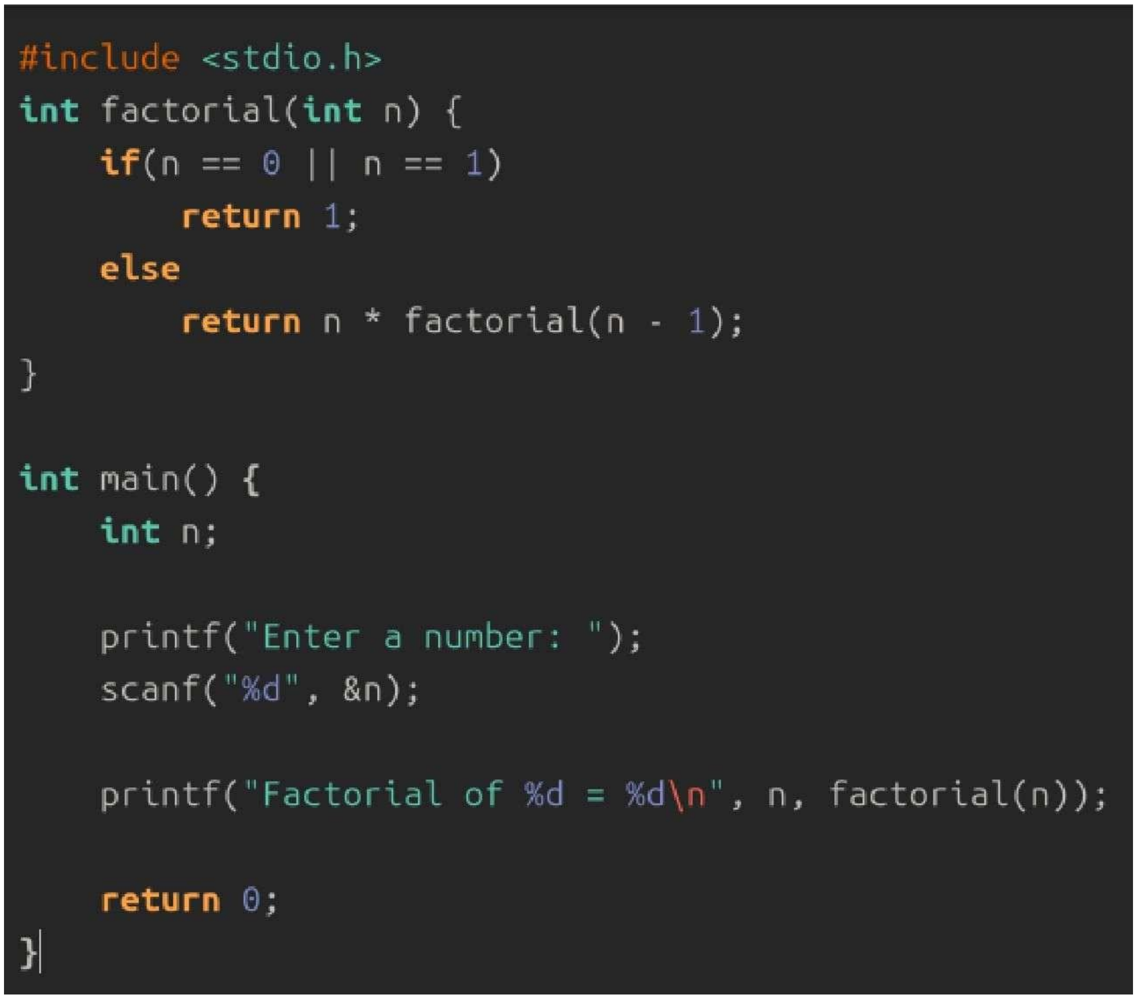
```
Enter the value of n: 3
Sum of cubes of first 3 natural numbers = 36
```

Space Complexity:

The space complexity of this program is **O(1)**, meaning constant space. Since it uses no recursion, no arrays, and no dynamic memory allocation, the memory used stays the same regardless of the value of *n*.

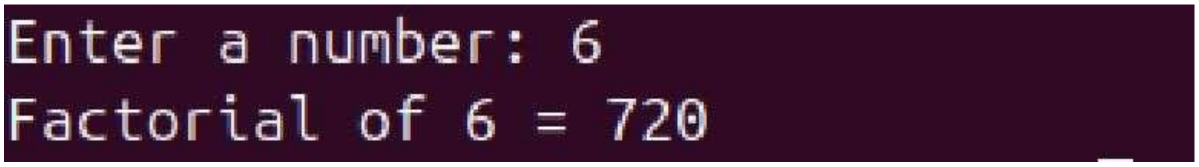
4) Write a program to find a factorial of given integer using recursion

Code:



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

int main() {
    int n;

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

    printf("Factorial of %d = %d\n", n, factorial(n));

    return 0;
}
```

Output:



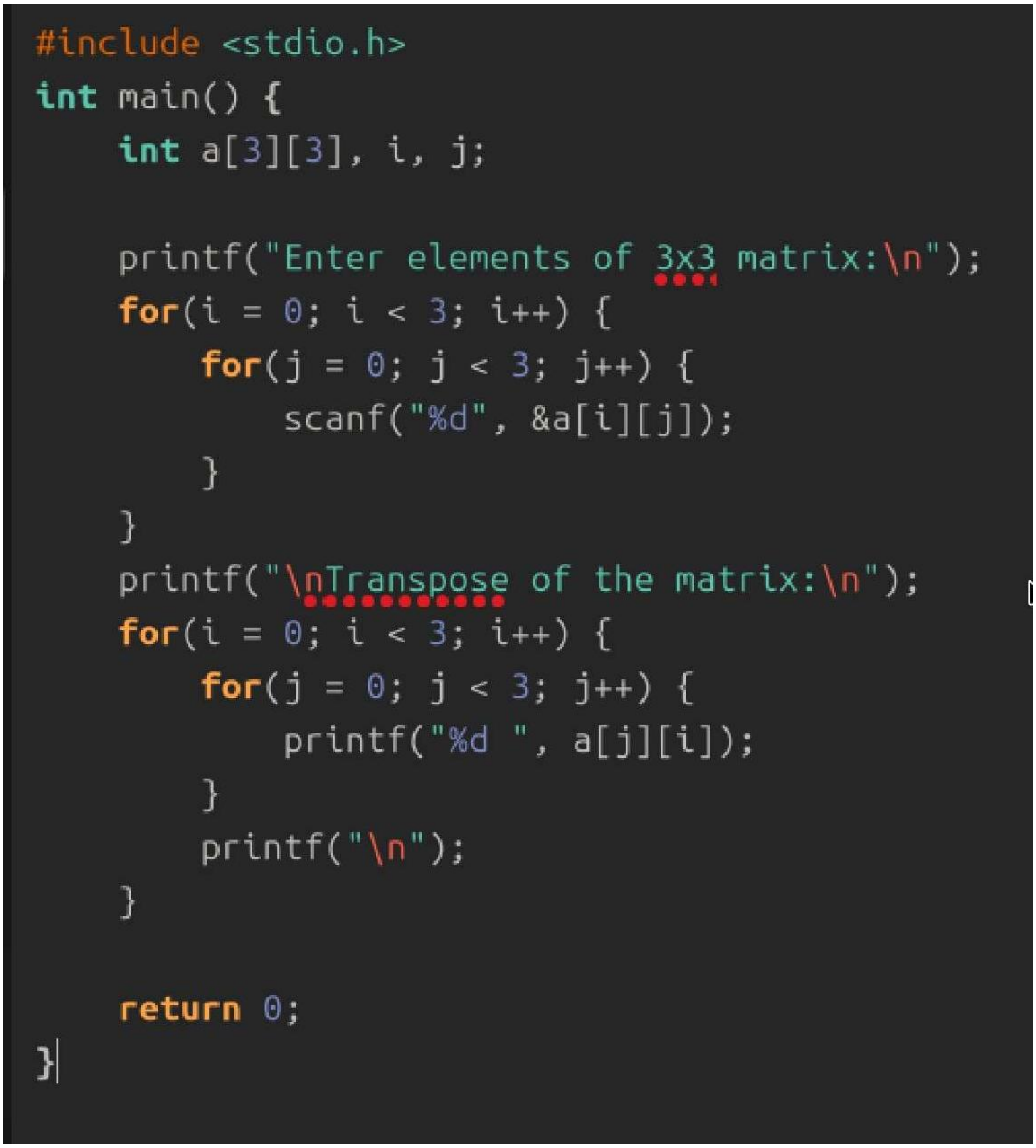
```
Enter a number: 6
Factorial of 6 = 720
```

Space Complexity:

The space complexity of this program is **O(n)** because the recursive function creates a new stack frame for each call. Even though no arrays or dynamic memory are used, the recursion depth depends on *n*, so the memory usage grows linearly.

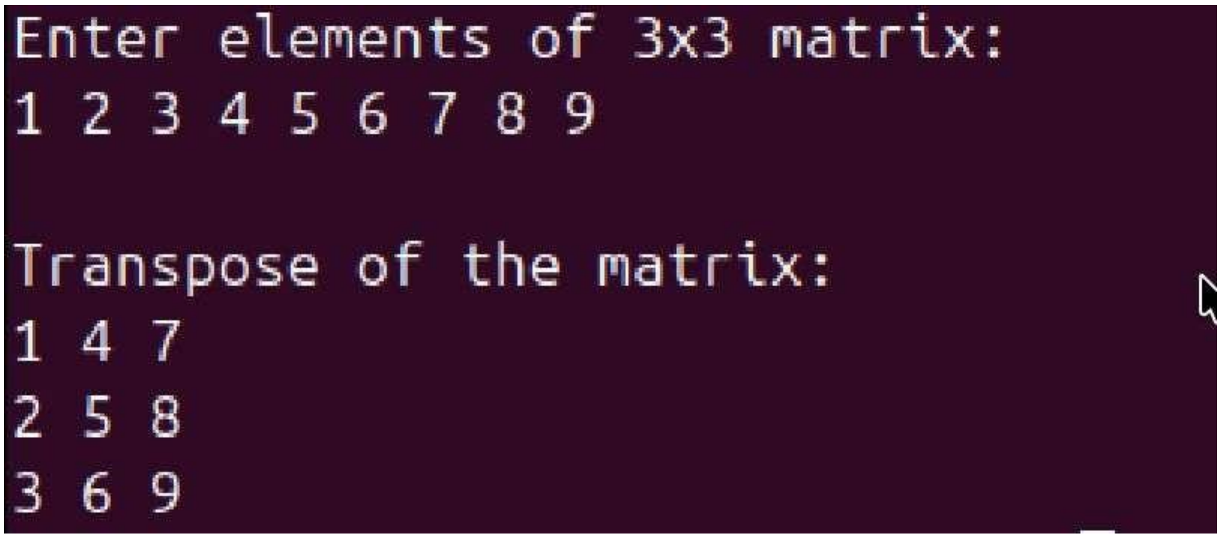
5) Write a program to find transpose of 3*3 matrix

Code:



```c
#include <stdio.h>
int main() {
    int a[3][3], i, j;

    printf("Enter elements of 3x3 matrix:\n");
    for(i = 0; i < 3; i++) {
        for(j = 0; j < 3; j++) {
            scanf("%d", &a[i][j]);
        }
    }
    printf("\nTranspose of the matrix:\n");
    for(i = 0; i < 3; i++) {
        for(j = 0; j < 3; j++) {
            printf("%d ", a[j][i]);
        }
        printf("\n");
    }

    return 0;
}
```

Output:



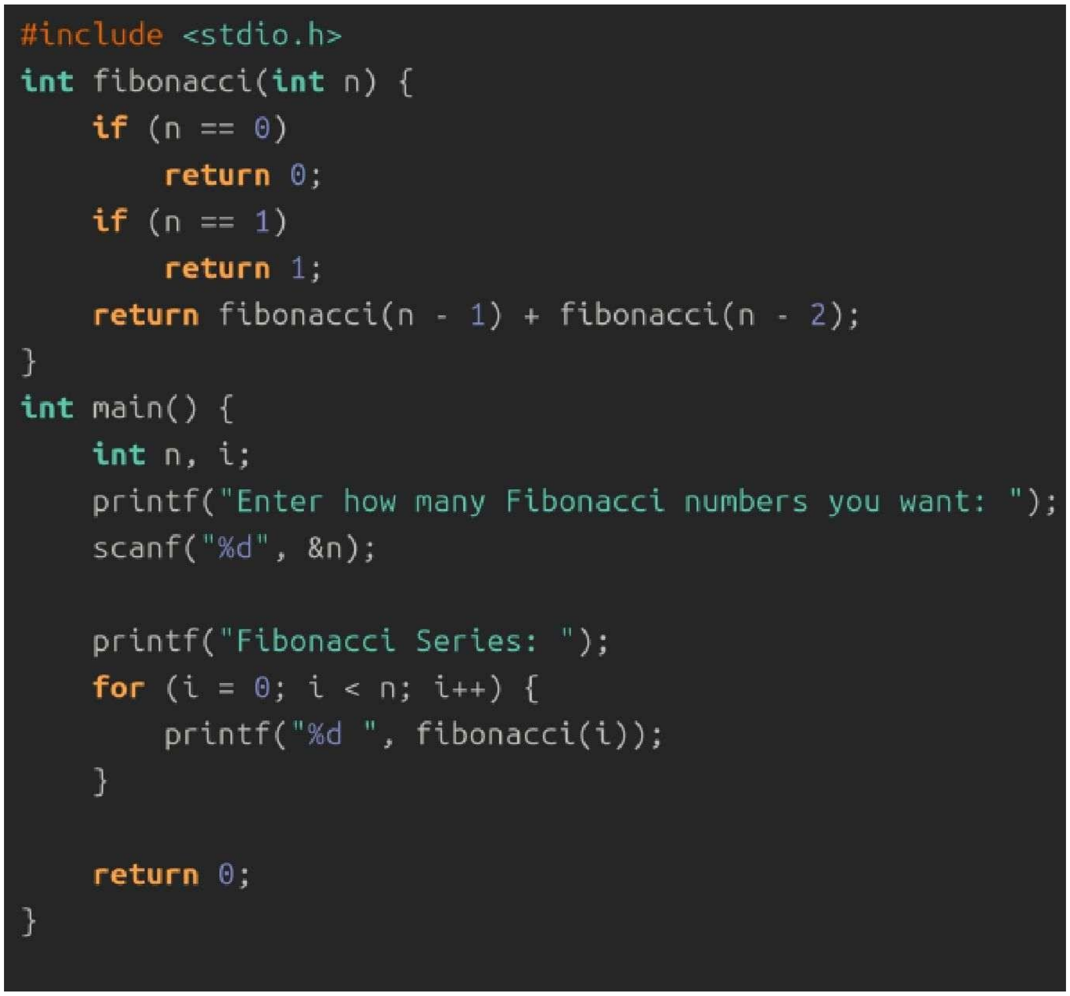
```
Enter elements of 3x3 matrix:
1 2 3 4 5 6 7 8 9

Transpose of the matrix:
1 4 7
2 5 8
3 6 9
```

Space Complexity:

The space complexity of this program is **O(1)** because the memory usage does not increase with the input size. It uses only a fixed amount of space, so the total space stays constant..

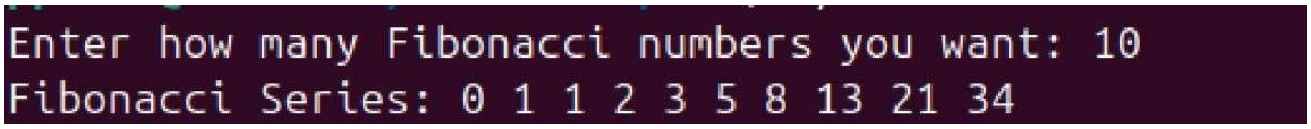
6) Write a program to find Fibonacci number at a given place

Code:



```c
#include <stdio.h>
int fibonacci(int n) {
    if (n == 0)
        return 0;
    if (n == 1)
        return 1;
    return fibonacci(n - 1) + fibonacci(n - 2);
}
int main() {
    int n, i;
    printf("Enter how many Fibonacci numbers you want: ");
    scanf("%d", &n);

    printf("Fibonacci Series: ");
    for (i = 0; i < n; i++) {
        printf("%d ", fibonacci(i));
    }

    return 0;
}
```

Output:



```
Enter how many Fibonacci numbers you want: 10
Fibonacci Series: 0 1 1 2 3 5 8 13 21 34
```

Space Complexity:

The space complexity of this program is **O(1)** because it uses no arrays, no recursion, and no dynamic memory allocation. It only uses a few fixed variables, so the memory needed does not change based on how many Fibonacci terms are printed.