23CSE211 Design and Analysis of Algorithms

Yugendra Senthil Kumar
CH.SC.U4CSE24252
CSE-C
30 November 2025

1.Write a program to calculate the sum of the first n natural numbers with a user defined function

```c
#include<stdio.h>
int sumN(int n){
    return n*(n+1)/2;
}
int main(){
    int n;
    scanf("%d",&n);
    printf("Sum:%d\n",sumN(n));
}
/*

The sum of the first n natural numbers is given by n*(n+1)/2
Space Complexity:O(c)constant space because constant number of variables.
Time Complexity: O(c)constant time because the formulaic calculation
doesn't scale with
input
*/
```

```
yugen@fountain:/mnt/c/Users/yugen/C files/Lab1$ gcc SumofN.c -o SumofN
yugen@fountain:/mnt/c/Users/yugen/C files/Lab1$ ./SumofN
4
Sum:10
yugen@fountain:/mnt/c/Users/yugen/C files/Lab1$
```

2.Write a program to calculate the sum of the squares of the first n natural numbers with a user defined function

```c
#include<stdio.h>
int sumOfSquares(int n){
    return n*(n+1)*(2*n+1)/6;
}
int main(){
    int n;
    scanf("%d",&n);
    printf("Sum:%d\n",sumOfSquares(n));
}
/*
The formula for the sum of the squares is given by n*(n+1)*(2*n+1)/6
Space Complexity O(c) constant space because there is a constant number of variables
Time Complexity O(c) constant time because the formulaic calculation doesn't scale with
input.
*/
```

```
yugen@fountain:/mnt/c/Users/yugen/C files/Lab1$ gcc sumofsquares.c -o sumofsquares
yugen@fountain:/mnt/c/Users/yugen/C files/Lab1$ ./sumofsquares
9
Sum:285
yugen@fountain:/mnt/c/Users/yugen/C files/Lab1$
```

3.Write a program to calculate the sum of cubes of the first n natural numbers.

```c
#include<stdio.h>
int main(){
    int n;
    scanf("%d",&n);
    printf("Sum:%d\n",(n*(n+1)/2)*(n*(n+1)/2));
}
/*
The formula for the sum of the first n natural cubes is the square of the sum of the
first n natural numbers
Space Complexity: O(c) constant space becuase constant number of variables
Time Complexity O(c) constant time beacuse the formulaic calcuation time doesn't
scale with input.
*/
```

```
yugen@fountain:/mnt/c/Users/yugen/C files/Lab1$ gcc sumofcubes.c -o sumofcubes
yugen@fountain:/mnt/c/Users/yugen/C files/Lab1$ ./sumofcubes
6
Sum:441
yugen@fountain:/mnt/c/Users/yugen/C files/Lab1$
```

4.Write a program to calculate the factorial of a natural number using a recursive function.

```c
#include<stdio.h>

int factorial(int num){
    if(num==2){
        return 2;
    }
    return num*(factorial(num-1));
}
int main(){
    int n;
    scanf("%d",&n);
    printf("%d\n",factorial(n));
}
/*
Time Complexity:O(n) linear time beacuse only one function call
Space Complexity:O(n) linear space beacuse of the recursion calling

This is done using recursion with a base case that returns 2(becuse 2! is
2).
Each time the function calls it self, it decrements num by 1, so the base
case
is always reached.
After that it will perform the multiplication, which then gives the
answer.
*/
```

```
yugen@fountain:/mnt/c/Users/yugen/C files/Lab1$ gcc factorial.c -o factorial
yugen@fountain:/mnt/c/Users/yugen/C files/Lab1$ ./factorial
4
24yugen@fountain:/mnt/c/Users/yugen/C files/Lab1$
```

5.Write a program to transpose a 3x3 matrix

```c
#include<stdio.h>

int main(){
    int matrix[3][3];
    int res[3][3];
    printf("Enter matrix elements\n");
    for(int i = 0;i<3;++i){
        for(int j = 0;j<3;++j){
            scanf("%d",&matrix[i][j]);
        }
    }
    for(int i = 0;i<3;++i){
        for(int j = 0;j<3;++j){
            res[i][j]=matrix[j][i];
            printf("%d\t",res[i][j]);
        }
        printf("\n");
    }
}


/*
2 nested for loops get the input
The other 2 nested for loops are responsible for
transposing(A(transpose)ij=Aji).
Time Complexity:O(c) constant time because the time taken to transpose a
3x3 matrix doesn't
depend on the elements of the matrix itself.
Space Complexity:O(c) constant space because it's always a 3x3 matrix, and
the elements of
the matrix have no effect on transposing.
*/
```

```
yugen@fountain:/mnt/c/Users/yugen/C files/Lab1$ gcc transposematrix.c -o transposematrix
yugen@fountain:/mnt/c/Users/yugen/C files/Lab1$ ./transposematrix
Enter matrix elements
1
2
3
4
5
6
7
8
9
1       4       7
2       5       8
3       6       9
yugen@fountain:/mnt/c/Users/yugen/C files/Lab1$
```

6.Write a program to print the fibonacci series upto a given number using a user-defined function.

```c
#include<stdio.h>

void fib(int n){
    int prev = 0;
    int curr = 1;
    for(int i = 0;i<n;++i){
        printf("%d\n",curr);
        int temp = curr;
        curr+=prev;
        prev = temp;
    }
}
int main(){
    int n;
    scanf("%d",&n);
    fib(n);
}
/*
Time Complexity:O(n) Linear time because of the for loop.
Space Complexity:O(c) constant space as there is a constant number of variables.
Curr holds the current fibonacci number, and prev holds the previous fibonacci number
that is used to increment curr.
A iterative loop is used to control how many times curr gets updated and printed.
*/
```

```
yugen@fountain:/mnt/c/Users/yugen/C files/Lab1$ gcc Fibo.c -o Fibo
yugen@fountain:/mnt/c/Users/yugen/C files/Lab1$ ./Fibo
4
1
1
2
3
yugen@fountain:/mnt/c/Users/yugen/C files/Lab1$
```