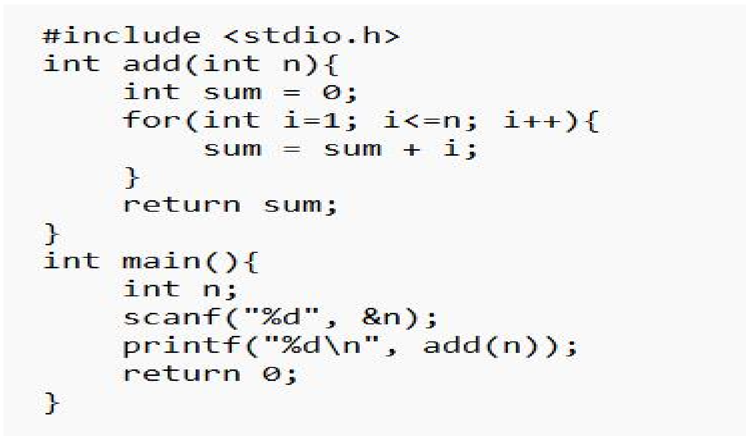
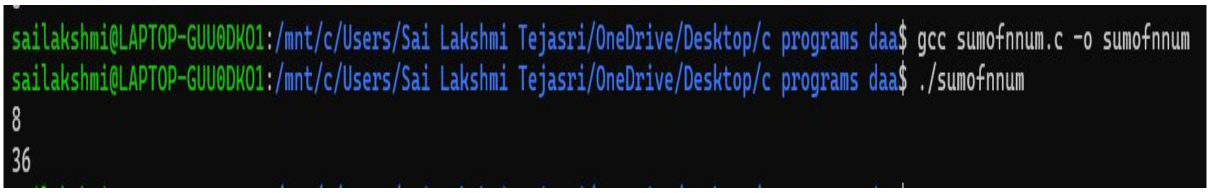
NAME: SAI LAKSHMI TEJASRI K

ROLL NO: CH.SC.U4CSE242423

1)Write a program to write sum of first n natural numbers using user defined function.



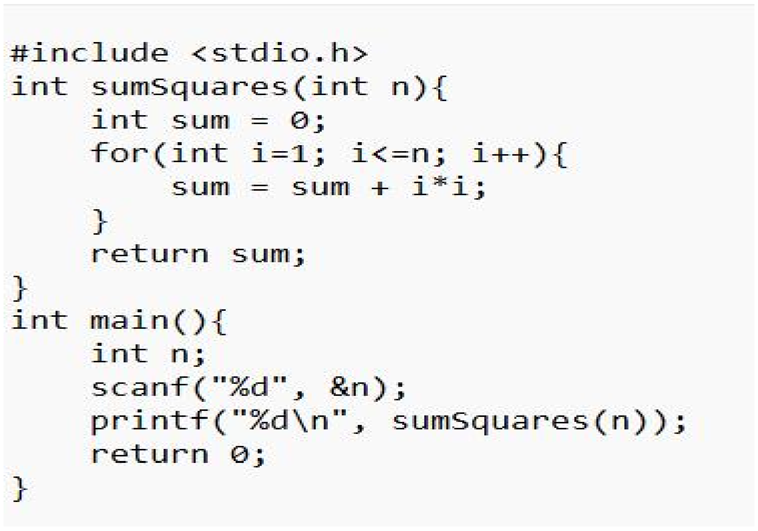
```c
#include <stdio.h>
int add(int n){
     int sum = 0;
     for(int i=1; i<=n; i++){
          sum = sum + i;
     }
     return sum;
}
int main(){
     int n;
     scanf("%d", &n);
     printf("%d\n", add(n));
     return 0;
}
```



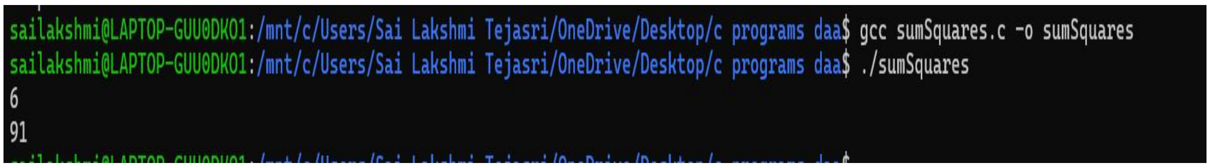
```
sailakshmi@LAPTOP-GUU0DK01:/mnt/c/Users/Sai Lakshmi Tejasri/OneDrive/Desktop/c programs daa$ gcc sumofnnum.c -o sumofnnum
sailakshmi@LAPTOP-GUU0DK01:/mnt/c/Users/Sai Lakshmi Tejasri/OneDrive/Desktop/c programs daa$ ./sumofnnum
8
36
```

Space Complexity: O(1)
Only sum, i, and n are used. No matter how big n is, memory stays the same.

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



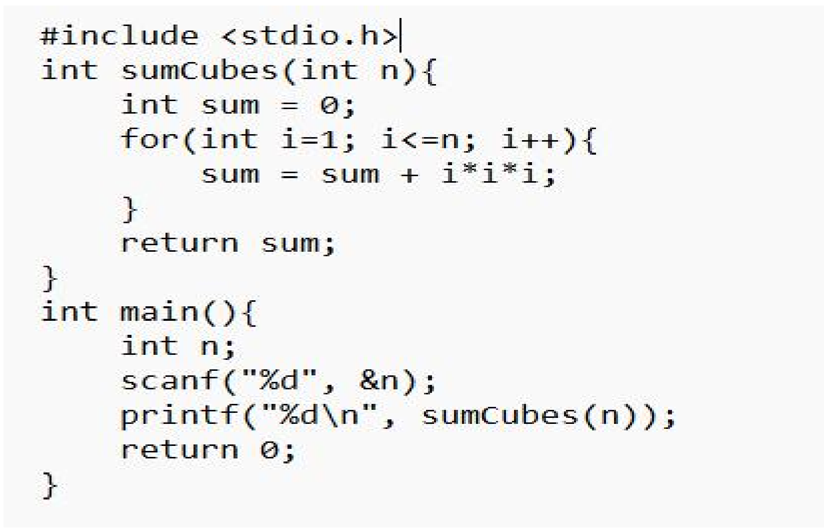
```c
#include <stdio.h>
int sumSquares(int n){
     int sum = 0;
     for(int i=1; i<=n; i++){
          sum = sum + i*i;
     }
     return sum;
}
int main(){
     int n;
     scanf("%d", &n);
     printf("%d\n", sumSquares(n));
     return 0;
}
```



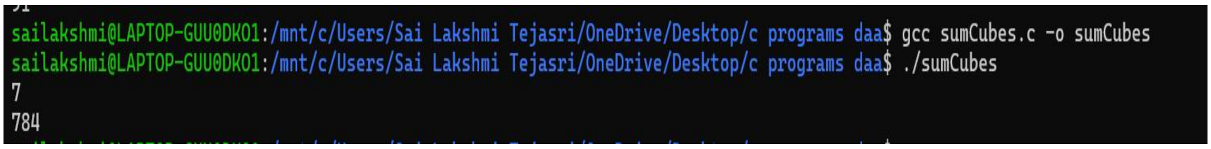
```
sailakshmi@LAPTOP-GUU0DK01:/mnt/c/Users/Sai Lakshmi Tejasri/OneDrive/Desktop/c programs daa$ gcc sumSquares.c -o sumSquares
sailakshmi@LAPTOP-GUU0DK01:/mnt/c/Users/Sai Lakshmi Tejasri/OneDrive/Desktop/c programs daa$ ./sumSquares
6
91
```

Space Complexity: O(1)
Only uses a few variables; memory does not increase with n.

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



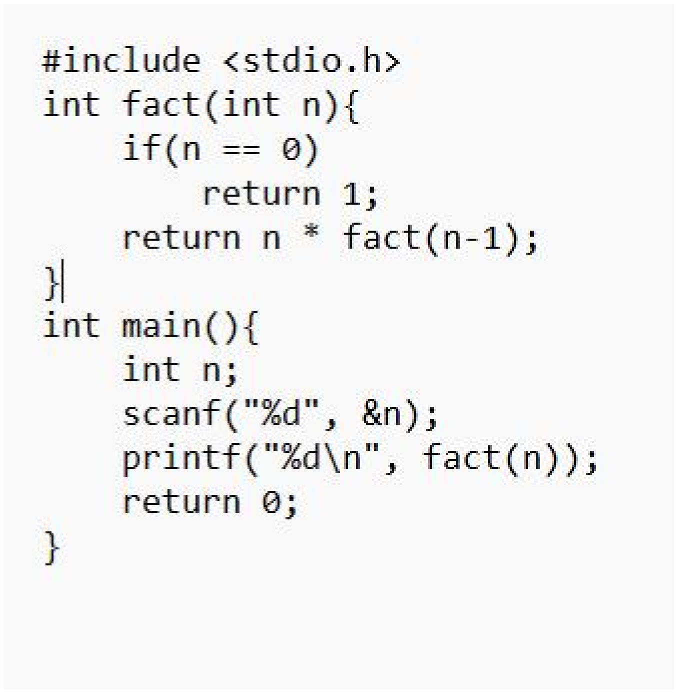
```
#include <stdio.h>
int sumCubes(int n){
    int sum = 0;
    for(int i=1; i<=n; i++){
        sum = sum + i*i*i;
    }
    return sum;
}
int main(){
    int n;
    scanf("%d", &n);
    printf("%d\n", sumCubes(n));
    return 0;
}
```



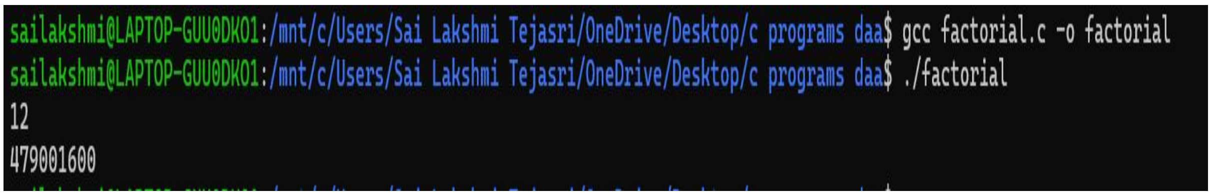
```
sailakshmi@LAPTOP-GUU0DK01:/mnt/c/Users/Sai Lakshmi Tejasri/OneDrive/Desktop/c programs daa$ gcc sumCubes.c -o sumCubes
sailakshmi@LAPTOP-GUU0DK01:/mnt/c/Users/Sai Lakshmi Tejasri/OneDrive/Desktop/c programs daa$ ./sumCubes
7
784
```

Space Complexity: O(1)
Because the program only uses a few variables (sum, i, n) and does not create any array or extra storage.
Memory stays the same no matter how big n is.

4)Write a program to write factorial of an given integer using recursion.



```
#include <stdio.h>
int fact(int n){
    if(n == 0)
        return 1;
    return n * fact(n-1);
}
int main(){
    int n;
    scanf("%d", &n);
    printf("%d\n", fact(n));
    return 0;
}
```

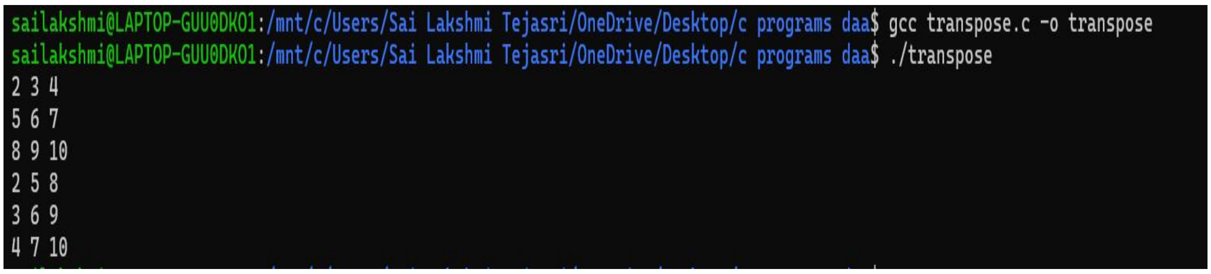


```
sailakshmi@LAPTOP-GUU0DK01:/mnt/c/Users/Sai Lakshmi Tejasri/OneDrive/Desktop/c programs daa$ gcc factorial.c -o factorial
sailakshmi@LAPTOP-GUU0DK01:/mnt/c/Users/Sai Lakshmi Tejasri/OneDrive/Desktop/c programs daa$ ./factorial
12
479001600
```

Space Complexity: O(n)
 Recursion means the function calls itself again and again.
Each call takes memory → so if n = 5, memory grows 5 levels deep.

5)Write a program for transposing a 3x3 matrix.

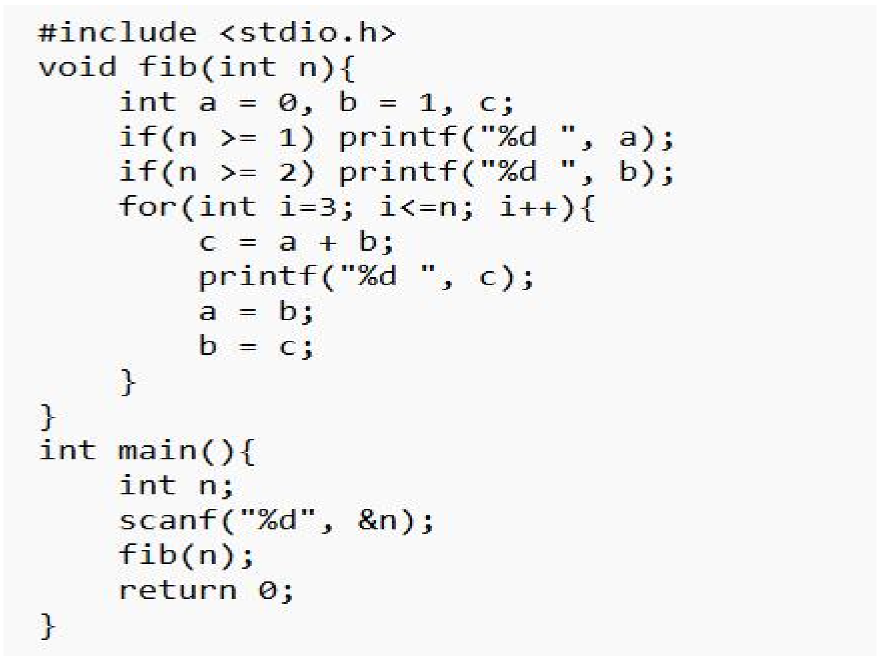
```
#include <stdio.h>
void transpose(int a[3][3]){
    for(int i=0; i<3; i++){
        for(int j=i+1; j<3; j++){
            int t = a[i][j];
            a[i][j] = a[j][i];
            a[j][i] = t;
        }
    }
}

int main(){
    int a[3][3];
    for(int i=0;i<3;i++)
        for(int j=0;j<3;j++)
            scanf("%d", &a[i][j]);
    transpose(a);
    for(int i=0;i<3;i++){
        for(int j=0;j<3;j++)
            printf("%d ", a[i][j]);
        printf("\n");
    }
    return 0;
}
```



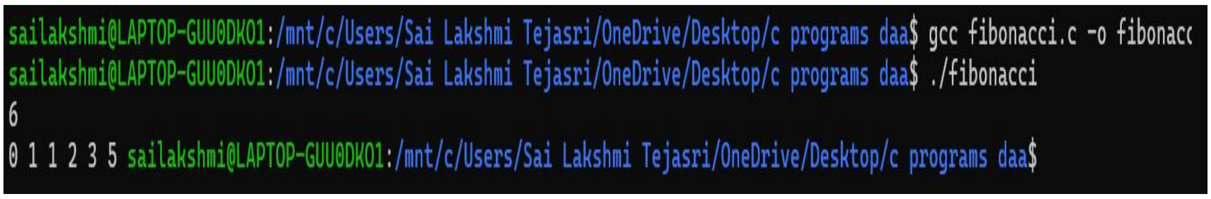
```
sailakshmi@LAPTOP-GUU0DK01:/mnt/c/Users/Sai Lakshmi Tejasri/OneDrive/Desktop/c programs daa$ gcc transpose.c -o transpose
sailakshmi@LAPTOP-GUU0DK01:/mnt/c/Users/Sai Lakshmi Tejasri/OneDrive/Desktop/c programs daa$ ./transpose
2 3 4
5 6 7
8 9 10
2 5 8
3 6 9
4 7 10
```

Space Complexity: No additional memory is used and every variable is fixed variable so the space complexity will be O(1) as same as first three problems.

6)Write a program to find Fibonacci series.



```c
#include <stdio.h>
void fib(int n){
    int a = 0, b = 1, c;
    if(n >= 1) printf("%d ", a);
    if(n >= 2) printf("%d ", b);
    for(int i=3; i<=n; i++){
        c = a + b;
        printf("%d ", c);
        a = b;
        b = c;
    }
}
int main(){
    int n;
    scanf("%d", &n);
    fib(n);
    return 0;
}
```



```
sailakshmi@LAPTOP-GUU0DK01:/mnt/c/Users/Sai Lakshmi Tejasri/OneDrive/Desktop/c programs daa$ gcc fibonacci.c -o fibonacc
sailakshmi@LAPTOP-GUU0DK01:/mnt/c/Users/Sai Lakshmi Tejasri/OneDrive/Desktop/c programs daa$ ./fibonacci
6
0 1 1 2 3 5 sailakshmi@LAPTOP-GUU0DK01:/mnt/c/Users/Sai Lakshmi Tejasri/OneDrive/Desktop/c programs daa$
```

Space Complexity: no memory is used and all the variables are fixed variables,same variables keeps changing but no external variable is taken.so the time complexity is O(1).