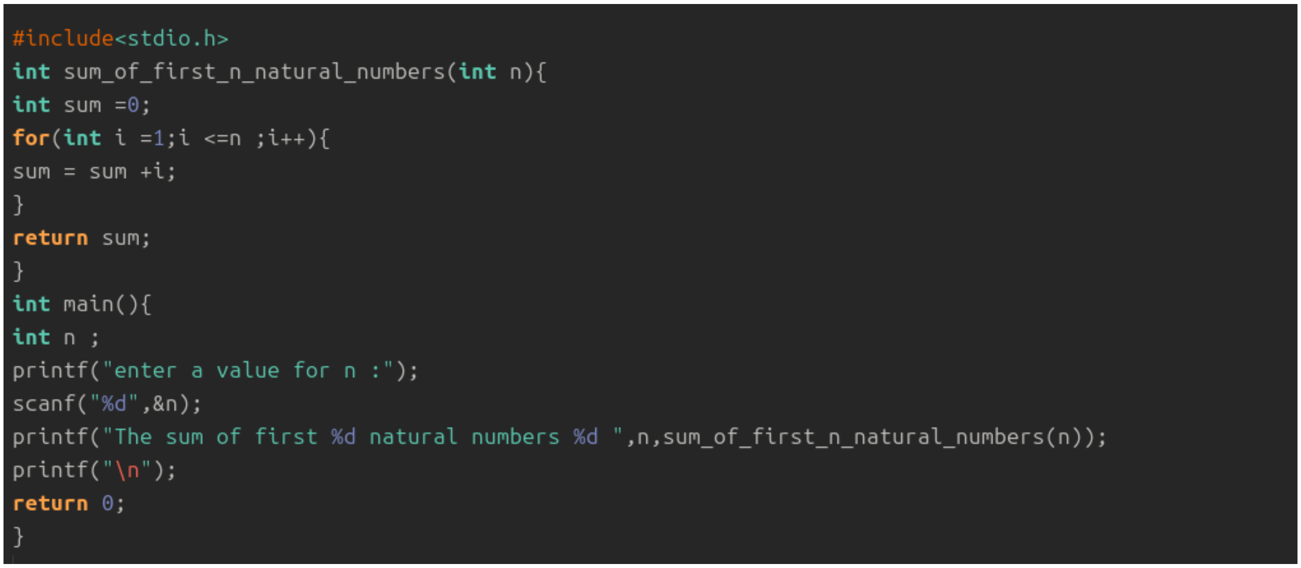
23CSE211-DESGIN AND ANALYSIS OF ALOGARTHIMS

LAB –1

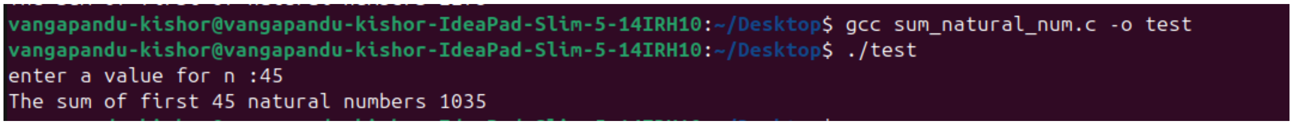
1.Write a program to find the sum of first n natural numbers using a use user defined function

C code:



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

OUTPUT:



```
vangapandu-kishor@vangapandu-kishor-IdeaPad-Slim-5-14IRH10:~/Desktop$ gcc sum_natural_num.c -o test
vangapandu-kishor@vangapandu-kishor-IdeaPad-Slim-5-14IRH10:~/Desktop$ ./test
enter a value for n :45
The sum of first 45 natural numbers 1035
```

SPACE COMPLEXITY:

• The loop uses only two variables (sum and i), and no extra memory grows

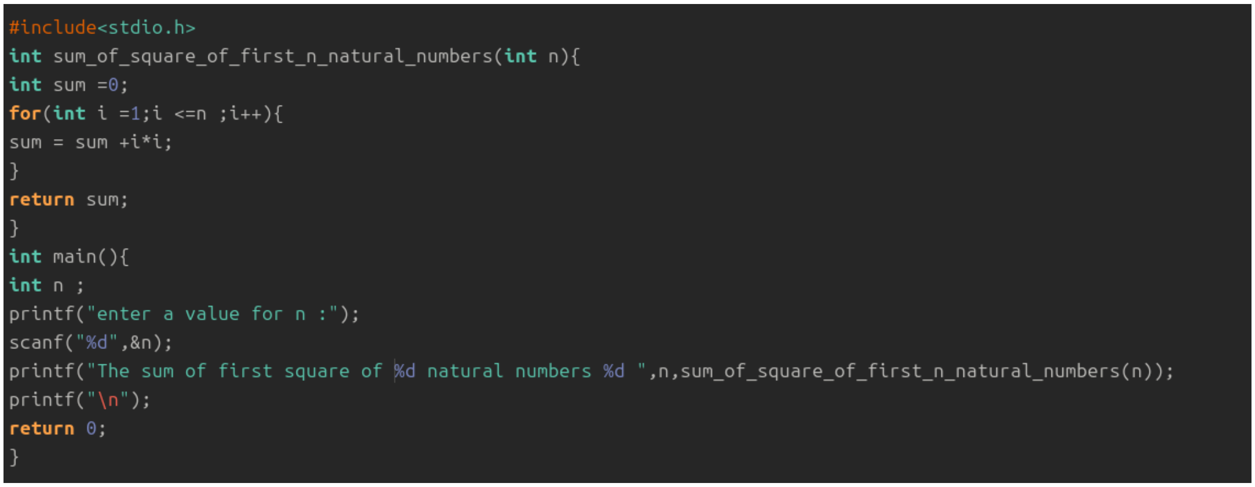
with n.

• So, the memory used stays constant no matter the input size.

• Therefore, the space complexity is O(1)

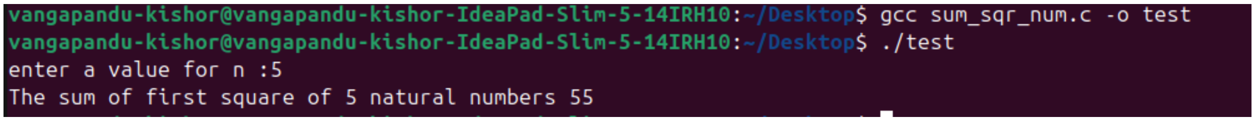
2. Write a program to find the sum of squares of first n natural numbers using a use user defined function

C code:



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

OUTPUT:



```
vangapandu-kishor@vangapandu-kishor-IdeaPad-Slim-5-14IRH10:~/Desktop$ gcc sum_sqr_num.c -o test
vangapandu-kishor@vangapandu-kishor-IdeaPad-Slim-5-14IRH10:~/Desktop$ ./test
enter a value for n :5
The sum of first square of 5 natural numbers 55
```

SPACE COMPLEXITY:

• The program uses only a few variables, and this number does not

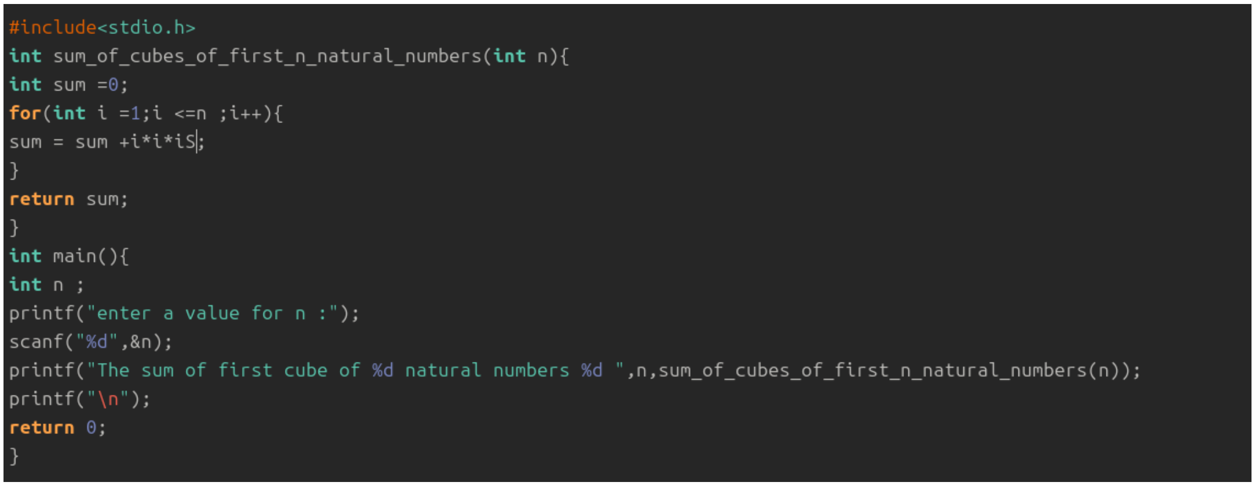
increase when n becomes bigger.

• Since the memory used stays the same all the time, the space

complexity is O(1).

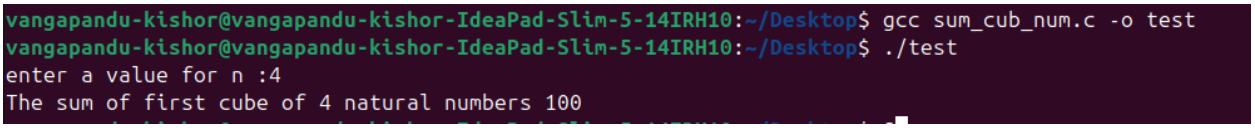
3. Write a program to find the sum of cubes of first n natural numbers using a use user defined function

C code:



```
#include<stdio.h>
int sum_of_cubes_of_first_n_natural_numbers(int n){
int sum =0;
for(int i =1;i <=n ;i++){
sum = sum +i*i*iS;
}
return sum;
}
int main(){
int n ;
printf("enter a value for n :");
scanf("%d",&n);
printf("The sum of first cube of %d natural numbers %d ",n,sum_of_cubes_of_first_n_natural_numbers(n));
printf("\n");
return 0;
}
```

OUTPUT:



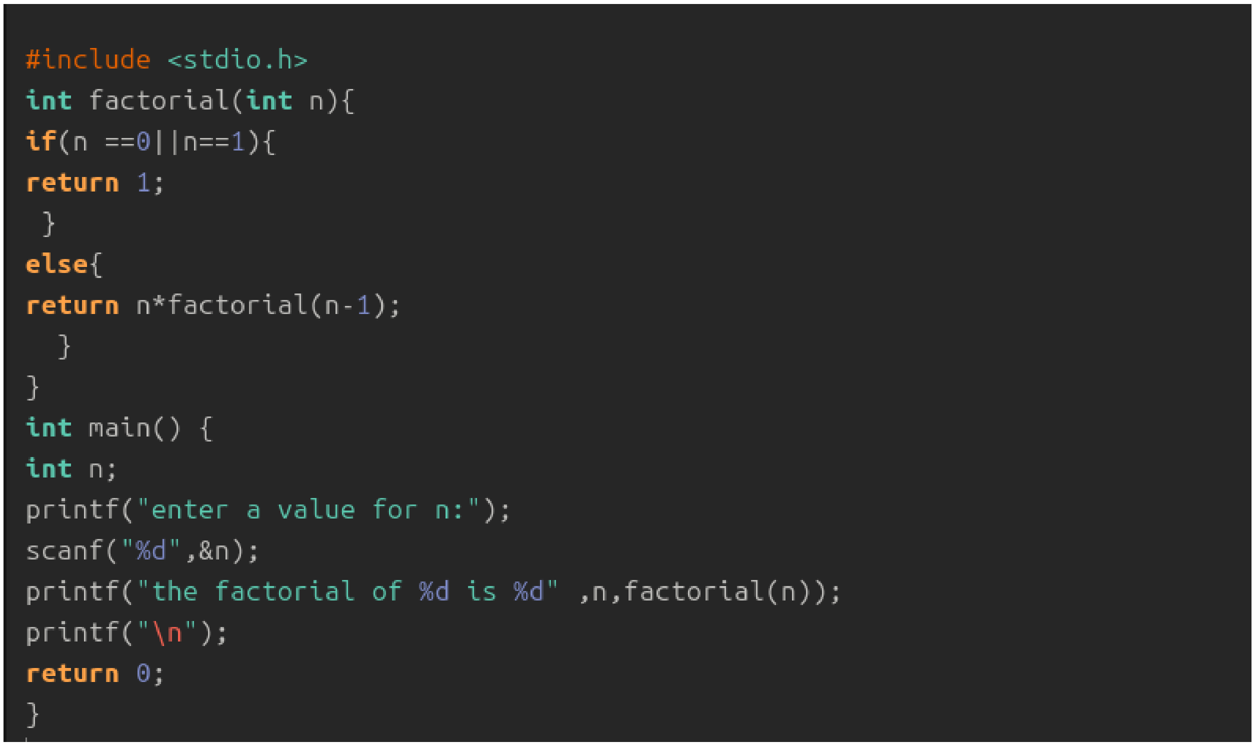
```
vangapandu-kishor@vangapandu-kishor-IdeaPad-Slim-5-14IRH10:~/Desktop$ gcc sum_cub_num.c -o test
vangapandu-kishor@vangapandu-kishor-IdeaPad-Slim-5-14IRH10:~/Desktop$ ./test
enter a value for n :4
The sum of first cube of 4 natural numbers 100
```

SPACE COMPLEXITY:

- The loop doesn't create new memory again and again — it just reuses the same variables.
- Because the memory doesn't grow when n grows, the space stays constant.
- So it is O(1) space.

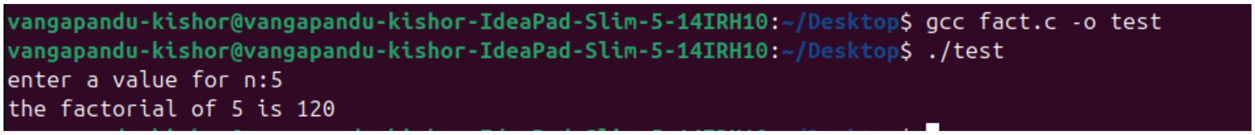
4. Write a program to find the factorial of a given number using recursion.

C 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 value for n:");
scanf("%d",&n);
printf("the factorial of %d is %d" ,n,factorial(n));
printf("\n");
return 0;
}
```

OUTPUT:



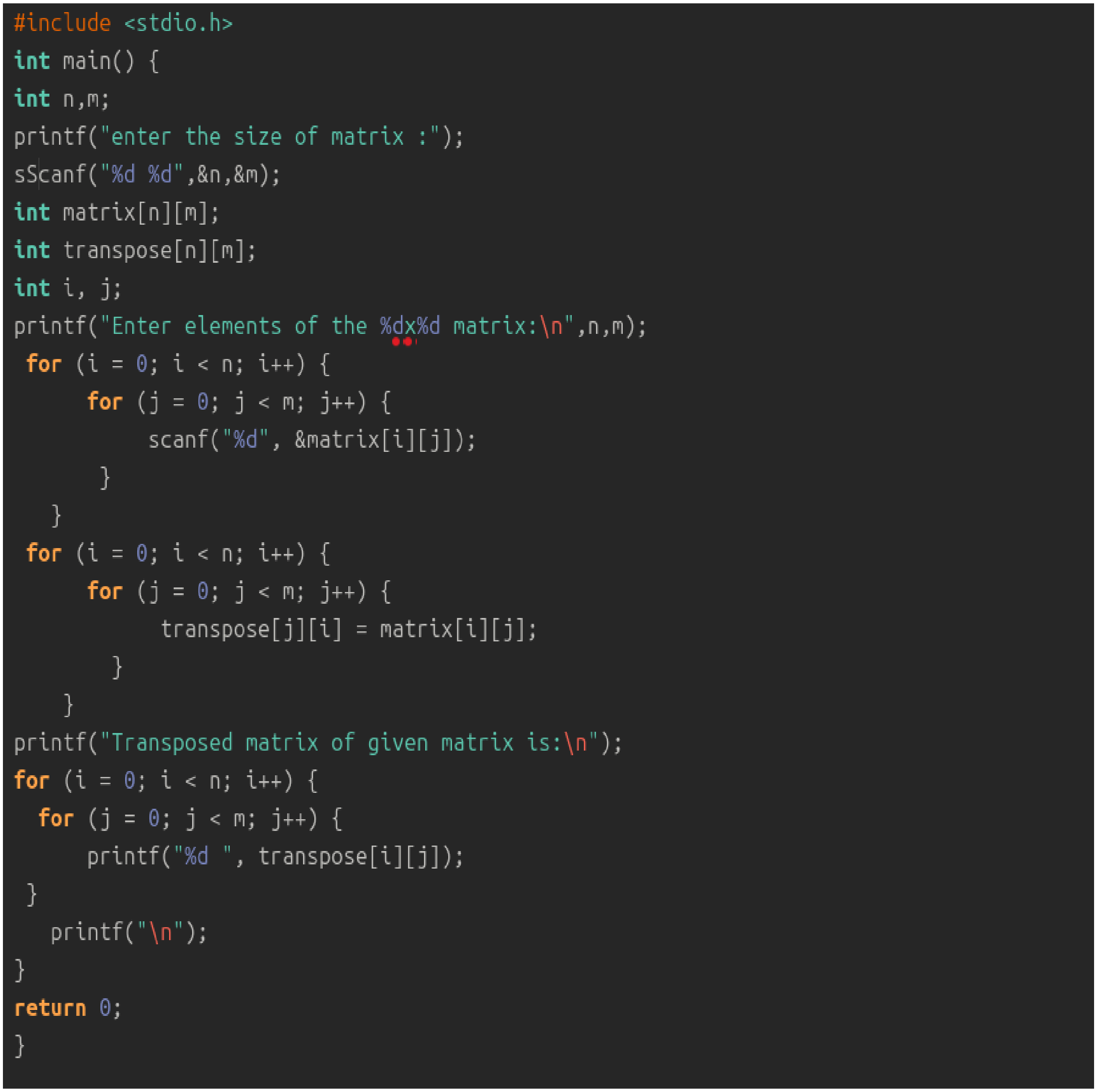
```
vangapandu-kishor@vangapandu-kishor-IdeaPad-Slim-5-14IRH10:~/Desktop$ gcc fact.c -o test
vangapandu-kishor@vangapandu-kishor-IdeaPad-Slim-5-14IRH10:~/Desktop$ ./test
enter a value for n:5
the factorial of 5 is 120
```

SPACE COMPLEXITY:

- The function calls itself many times, and each call uses some memory.
- More calls mean more memory is used, so memory increases with n.
- That's why the space complexity is O(n).

5. Write a program to transpose a 3x3 matrix.

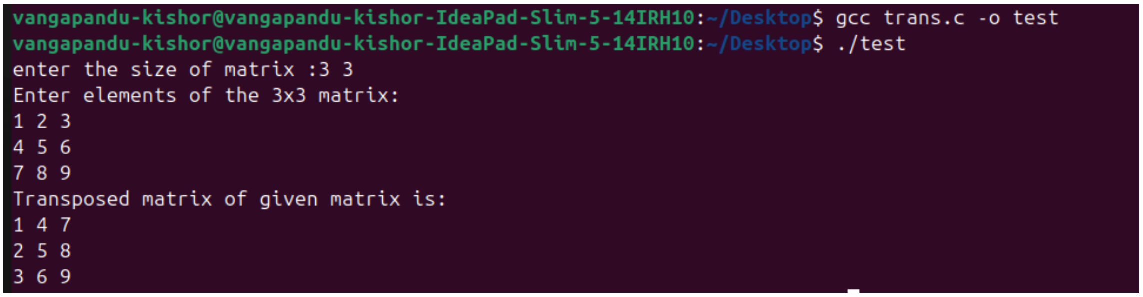
C code:



```c
#include <stdio.h>
int main() {
int n,m;
printf("enter the size of matrix :");
sScanf("%d %d",&n,&m);
int matrix[n][m];
int transpose[n][m];
int i, j;
printf("Enter elements of the %dx%d matrix:\n",n,m);
 for (i = 0; i < n; i++) {
      for (j = 0; j < m; j++) {
           scanf("%d", &matrix[i][j]);
       }
   }
 for (i = 0; i < n; i++) {
      for (j = 0; j < m; j++) {
            transpose[j][i] = matrix[i][j];
        }
    }
printf("Transposed matrix of given matrix is:\n");
for (i = 0; i < n; i++) {
  for (j = 0; j < m; j++) {
      printf("%d ", transpose[i][j]);
 }
   printf("\n");
}
return 0;
}
```

\

OUTPUT:



```
vangapandu-kishor@vangapandu-kishor-IdeaPad-Slim-5-14IRH10:~/Desktop$ gcc trans.c -o test
vangapandu-kishor@vangapandu-kishor-IdeaPad-Slim-5-14IRH10:~/Desktop$ ./test
enter the size of matrix :3 3
Enter elements of the 3x3 matrix:
1 2 3
4 5 6
7 8 9
Transposed matrix of given matrix is:
1 4 7
2 5 8
3 6 9
```

SPACE COMPLEXITY:

•for a 3×3 array, which is fixed in size and does not

Grow.

• You only use a few extra variables for loops.

• So ,the space complexity is O(1)

OR

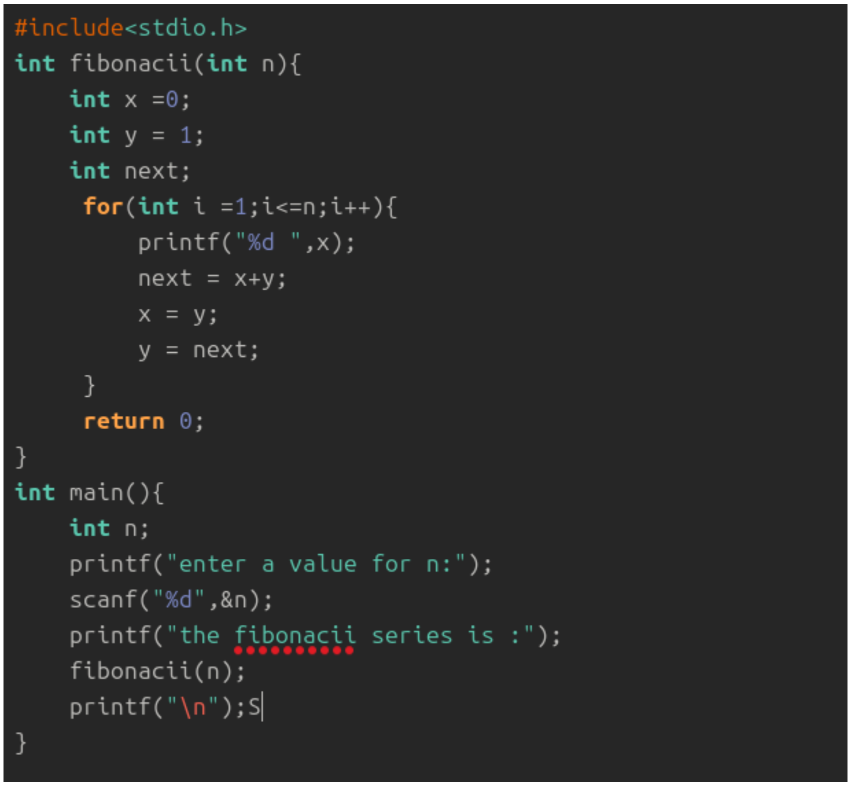
Here,n and m are user inputs (variable-size matrix)

- matrix uses **O(nm)** space
- transpose uses **O(nm)** space
- **Space complexity = O(n × m)**

For a **square matrix** (n = m): **Space complexity = O(n).**

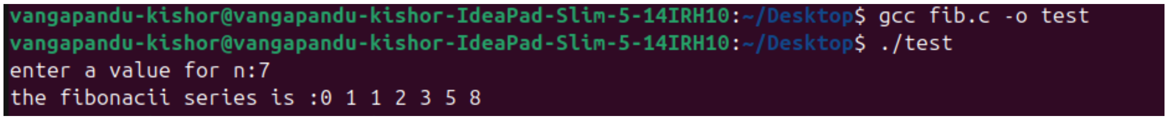
6. Write a program to find the Fibonacci series.

C code:



```c
#include<stdio.h>
int fibonacii(int n){
    int x =0;
    int y = 1;
    int next;
     for(int i =1;i<=n;i++){
         printf("%d ",x);
         next = x+y;
         x = y;
         y = next;
     }
     return 0;
}
int main(){
    int n;
    printf("enter a value for n:");
    scanf("%d",&n);
    printf("the fibonacii series is :");
    fibonacii(n);
    printf("\n");S
}
```

Output:



```
vangapandu-kishor@vangapandu-kishor-IdeaPad-Slim-5-14IRH10:~/Desktop$ gcc fib.c -o test
vangapandu-kishor@vangapandu-kishor-IdeaPad-Slim-5-14IRH10:~/Desktop$ ./test
enter a value for n:7
the fibonacii series is :0 1 1 2 3 5 8
```

SPACE COMPLEXITY:

- The program only uses a few variables (x, y, next)and these do

not increase when n increases.

- No extra memory grows with the loop.
- So ,the space complexity is O(1)