SCHOOL OF COMPUTING

DHEERAJ VAMSI KRISHNA

CH.SC.U4CSE24127

WEEK 1

Design and Analysis of Algorithm(23CSE211)

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

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

Output:

```
amma@amma24:~/dheeraj$ ./1
Enter the value for n
5
15amma@amma24:~/dheeraj$
```

Space Complexity:

Space Complexity O(1) 3 variables

Justification:

The program only uses a few variables (n, sum, i).
No extra memory like arrays or big data is used.
So the memory used stays the same for any value of *n*.
That's why the space complexity is O(1).

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

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

Output:

```
amma@amma24:~/dheeraj$ ./2
enter a number n:
6
the sum of 6 natural numbers is 91amma@amma24:~/dheeraj$
```

Space complexity

The program only uses a few integer variables (n, i, sum).
There are no arrays or extra memory used.
So the memory does not change even if *n* becomes big.
That's why the space complexity is O(1).

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

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

Output:

```
amma@amma24:~/dheeraj$ ./3
enter a numbers n:
4
the sum of 4 natural numbers is 100amma@amma24:~/dheeraj$
```

Space complexity:

The program only uses three integer variables: n, i, and sum.
No arrays or extra memory are used.
So the memory stays the same for any value of *n*.
That's why the space complexity is O(1).

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

```c
#include <stdio.h>
int fact(int n){
if (n==0||n==1){
return 1;
}
else{
 return n*fact(n-1);
 }
 }
int main (){
int n;
printf("enter the number:");
scanf("%d",&n);
int result =fact(n);
printf("%d",result);
}
```

Output:

```
amma@amma24:~/dheeraj$ gcc -o 4 4.c
amma@amma24:~/dheeraj$ ./4
enter the number:4
24amma@amma24:~/dheeraj$
```

Space complexity:

This program uses recursion, and each function call is stored in the stack.
For fact(n) it will call fact(n-1), fact(n-2), … until 1.
So there will be n function calls in memory.
Because of this, the space grows as *n* increases.
So the space complexity is O(n).

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

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

    printf("Enter 9 elements of the 3x3 matrix:\n");
    for(i = 0; i < 3; i++){
        for(j = 0; j < 3; j++){
            scanf("%d", &a[i][j]);
        }
    }
    for(i = 0; i < 3; i++){
        for(j = 0; j < 3; j++){
            t[i][j] = a[j][i];
        }
    }
    printf("Transpose of the matrix:\n");
    for(i = 0; i < 3; i++){
        for(j = 0; j < 3; j++){
            printf("%d ", t[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

Output:

```
PS D:\c programming> ./e.exe
Enter 9 elements of the 3x3 matrix:
3 4 5
4 6 7
9 7 2
Transpose of the matrix:
3 4 9
4 6 7
5 7 2
PS D:\c programming>
```

Space complexity:

The matrix size is fixed (3×3).
We always use the same amount of memory for the arrays and the loop variables.
So the memory does not increase with input size.

Therefore, the space complexity is O(1).

6)Write a program to find Fibonacci series

```c
#include <stdio.h>
int main() {
    int n, a = 0, b = 1, c, i;
    printf("Enter how many terms: ");
    scanf("%d", &n);
    printf("Fibonacci Series:\n");
    for(i = 1; i <= n; i++){
        printf("%d ", a);
        c = a + b;
        a = b;
        b = c;
    }

    return 0;
}
```

Output:

```
PS D:\c programming> gcc e.c -o e
PS D:\c programming> ./e.exe
Enter how many terms: 8
Fibonacci Series:
0 1 1 2 3 5 8 13
PS D:\c programming>
```

The program only uses a few variables (a, b, c, n, i).
There are no arrays or extra memory used.
So the memory used stays the same for any number of terms.

Therefore, the space complexity is O(1).