DAA SPACE COMPLEXITY PRACTICE PROBLEMS

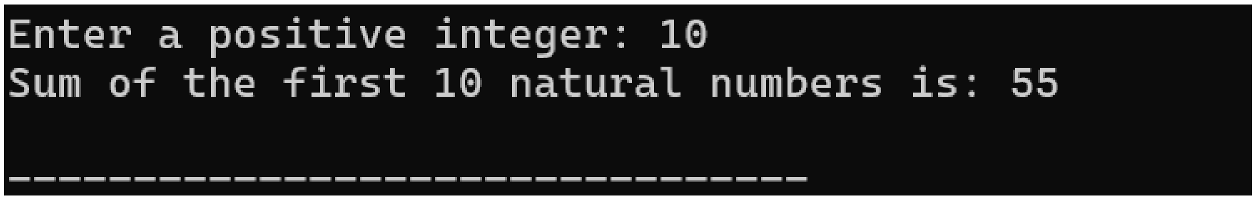
1.write a program to find sum of first n natural numbers using user defined function.

Code:

```
#include <stdio.h>
int findsum(int n);
int main() {
    int n, result;
    printf("Enter a positive integer: ");
    scanf("%d", &n);
    result = findsum(n);
    printf("Sum of the first %d natural numbers is: %d\n", n, result);
    return 0;
}
int findsum(int n) {
    int sum = 0;
    int i;

    for(i = 1; i <= n; i++) {
        sum += i;
    }
    return sum;
}
```

Output:



```
Enter a positive integer: 10
Sum of the first 10 natural numbers is: 55

--------------------------------
```

Space complexity: The space complexity of this program is O(1) - constant space . Because the program uses a **fixed number of variables**, independent of the input size, its space usage does not grow.

2. Write a program to find Sum of Square of first N natural numbers.

Code:

```
#include <stdio.h>
int sumofsquares(int n);
int main() {
int n, result;
int i;
printf("Enter a positive integer: ");
scanf("%d", &n);
result = sumofsquares(n);
printf("Sum of square of the first %d natural numbers is: %d\n", n, result);
return 0;
}
int sumofsquares(int n) {
int sum = 0;
int i;
```

for(i = 1; i<=n; i++) {

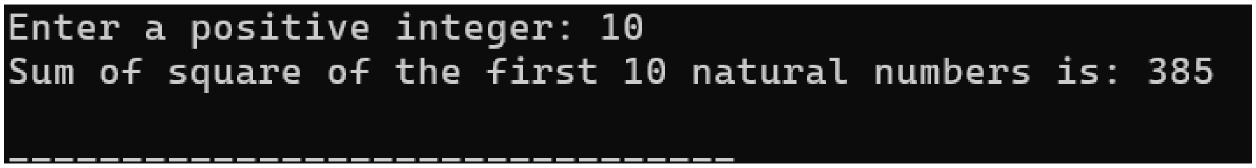
sum+=i*i;

}

return sum;

}

Output:



```
Enter a positive integer: 10
Sum of square of the first 10 natural numbers is: 385
```

Space complexity: The space complexity of this program is O(1) - constant space . Because there is no recursion, no arrays , no dynamic memory allocation , the amount of memory used **does not depend on n**.

3. Write a program to find Sum of Cubes of first N natural numbers.

Code:

#include <stdio.h>

int sumofcubes(int n);

int main() {

int n, result;

int i;

printf("Enter a positive integer: ");

scanf("%d", &n);

result = sumofcubes(n);

printf("Sum of cubes of the first %d natural numbers is: %d\n", n, result);

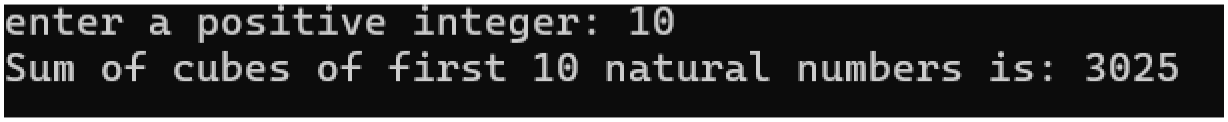
return 0;

}

int sumofcubes(int n) {

```
int sum = 0;
int i;
for(i = 1; i<=n; i++) {
sum+=i*i*i
}
return sum;
}
```

Output:



```
enter a positive integer: 10
Sum of cubes of first 10 natural numbers is: 3025
```

Space complexity: The space complexity of this program is O(1) - constant space . Because there is no recursion, no arrays , no dynamic memory allocation , the amount of memory used **does not depend on n**.

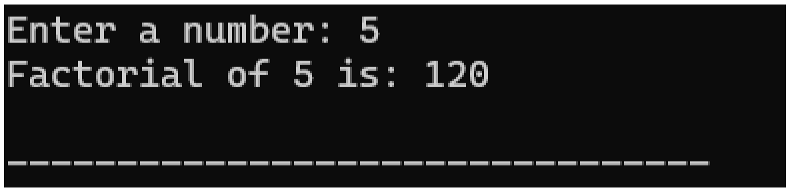
4. Write a program to find factorial of a number using recursion.

Code:

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

```
    printf("Enter a number: ");
    scanf("%d", &n);
    if(n < 0) {
        printf("Factorial is not defined for negative numbers.\n");
    } else {
        printf("Factorial of %d is: %lld\n", n, factorial(n));
    }
    return 0;
}
```

Output:



```
Enter a number: 5
Factorial of 5 is: 120

--------------------------------
```

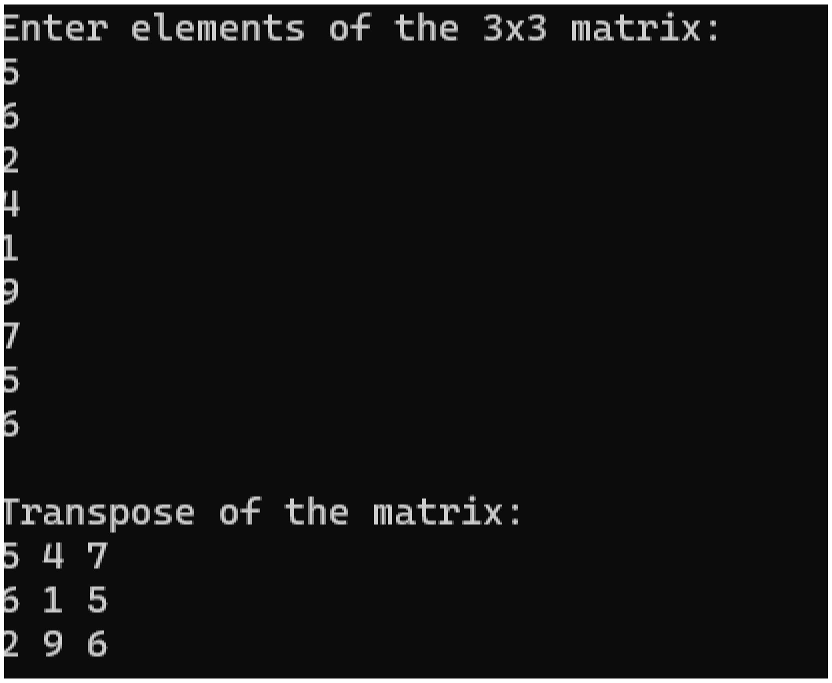
Space complexity: The space complexity of this problem is O(n). There are no arrays or dynamic memory allocation, but the recursive call stack makes the memory usage **depend on n**, resulting in **O(n)** space complexity.

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

Code:

```
#include <stdio.h>
int main() {
    int a[3][3], trans[3][3];
    int i, j;
    printf("Enter 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++) {
            trans[j][i] = a[i][j];
        }
    }
    printf("\nTranspose of the matrix:\n");
    for(i = 0; i < 3; i++) {
        for(j = 0; j < 3; j++) {
            printf("%d ", trans[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

Output:



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

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

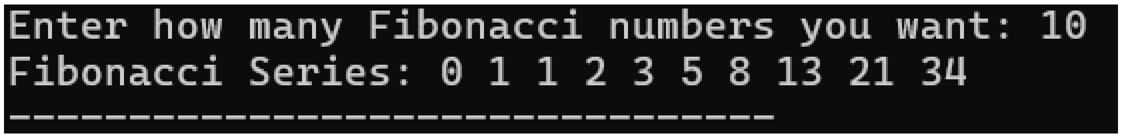
<u>**Space complexity:**</u> The space complexity of this program is O(1) - constant space. Because the memory used does **not grow with input size**, the total space remains constant.

6. Write a program to find fibonacci numbers of a given number.

<u>**Code:**</u>

```
#include <stdio.h>
int main() {
    int n, a = 0, b = 1, c, i;
    printf("Enter how many Fibonacci numbers you want: ");
    scanf("%d", &n);
    printf("Fibonacci Series: ");
    for(i = 1; i <= n; i++) {
        printf("%d ", a);
        c = a + b;
        a = b;
        b = c;
    }
    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) - constant space. There are **no arrays**, **no recursion**, and **no dynamic memory allocation**, so the memory usage does **not depend on the number of Fibonacci terms printed**.