

DAA

KURRA . VENKATA GOKUL

CH.SC.U4CSE24121

CSE-B

1.FACTORIAL OF GIVEN NUMBER USING RECURSIVE FUNCTION

CODE:

```
#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 N value :");
scanf("%d",&N);
int sum=fact(N);
printf(" the factorial of given number:%d\n",sum);
return 0;
}
```

OUTPUT:

```
C:\Users\kurra\OneDrive\Desktop\SEM IV\DAA>.\a
enter the N value :5
the factorial of given number:120

C:\Users\kurra\OneDrive\Desktop\SEM IV\DAA>|
```

SPACE COMPLEXITY:

Space Complexity = $O(N)$

The Recursive function excuting the N times at same time.

2.FIBONOCCHI SERIES

CODE:

```
File Edit View

#include <stdio.h>

int main() {
    int N;
    int f = 0, f1 = 1, fib, i;

    printf("Enter how many terms: ");
    scanf("%d", &N);

    printf("Fibonacci Series: ");

    if (N >= 1)
        printf("%d ", f);
    if (N >= 2)
        printf("%d ", f1);

    for (i = 3; i <= N; i++) {
        fib = f + f1;
        printf("%d ", fib);

        f = f1;
        f1 = fib;
    }

    return 0;
}
```

OUTPUT:

```
C:\Users\kurra\OneDrive\Desktop\SEM IV\DAA>gcc -o fib fib.c  
C:\Users\kurra\OneDrive\Desktop\SEM IV\DAA>.\fib  
Enter how many terms: 5  
Fibonacci Series: 0 1 1 2 3  
C:\Users\kurra\OneDrive\Desktop\SEM IV\DAA>|
```

Space complexity :

In the iterative Fibonacci program, we only use a **fixed number of variables**:

F, f1, fib, loop variable l, input N

No matter how large N is, the program does **not** create any arrays, no extra data structures, and does not grow memory as N increases.

O(1) — Constant Space

3.SUM OF N NUMBERS

CODE:

```
File    Edit    View

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

OUTPUT:

```
C:\Users\kurra\OneDrive\Desktop\SEM IV\DAA>.s
enter the N number :5
the sum of N numbers is 15:
C:\Users\kurra\OneDrive\Desktop\SEM IV\DAA>
C:\Users\kurra\OneDrive\Desktop\SEM IV\DAA>|
```

SPACE COMPLEXITY:

Your program uses only a few variables:

I,Sum,N,total

It does not use any array, list, or extra memory that grows when N becomes large.

$O(1)$ – CONSTANT SPACE

4.TRANSPOSE A MATRIX

CODE:

```
#include<stdio.h>
int main()
{
    int i,j,arr[3][3];
    //input matrix values
    for(i=0;i<3;i++)
    {
        for( j=0;j<3;j++)
        {
            scanf("%d",&arr[i][j]);
        }
    }

    //transpose
    for( i=0;i<3;i++)
    {
        for( j=1;j<3;j++)
        {
            int temp = arr[i][j];
            arr[i][j] = arr[j][i];
            arr[j][i] = temp;
        }
    }

    printf("Transpose of the matrix:\n");
    for( i = 0; i < 3; i++) {
        for(j = 0; j < 3; j++) {
            printf("%d ", arr[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

OUTPUT:

```
C:\Users\kurra\OneDrive\Desktop\SEM IV\DAA>gcc -o mat mat.c
C:\Users\kurra\OneDrive\Desktop\SEM IV\DAA>.\mat
1
2
3
7
8
9
4
5
6
Transpose of the matrix:
1 7 4
2 8 9
3 5 6
```

SPACE COMPLEXITY:

The matrix size is fixed.

Only a few variables are used.

Memory does not grow.

$O(1)$ -- SPACE COMPLEXITY

5. SUM OF THE SQUARES OF N NUMBERS

CODE:

```

#include<stdio.h>
int main()
{
int N,sum=0,i,p;
printf("enter the number of Natural Number N:");
scanf("%d",&N);
for(i=1;i<N;i++)
{ p=N*N;
sum=sum+p;
}
printf("the sum of squares of a N Number%d",sum);
}

```

OUTPUT:

```

C:\Users\kurra\OneDrive\Desktop\SEM IV\DAA>gcc -o sum sums.c
C:\Users\kurra\OneDrive\Desktop\SEM IV\DAA>.\sum
enter the number of Natural Number N:5
the sum of squares of a N Number100

```

SPACE COMPLEXITY:

variables N, sum, i, and p.

No arrays, no extra memory.

Memory does not increase with N.

$O(1)$.

6.SUM OF THE CUBES OF N NUMBERS

CODE:

```
#include<stdio.h>
int main()
{
    int N,sum=0,i,p;
    printf("enter the number of Natural Number N:");
    scanf("%d",&N);
    for(i=1;i<N;i++)
    { p=N*N*N;
      sum=sum+p;
    }
    printf("the sum of squares of a N Number%d",sum);
}
```

OUTPUT:

```
C:\Users\kurra\OneDrive\Desktop\SEM IV\DAA>gcc -o suml sumc.c
C:\Users\kurra\OneDrive\Desktop\SEM IV\DAA>.\suml
enter the number of Natural Number N:5
the sum of squares of a N Number500
```

SPACE COMPLEXITY:

variables N, sum, i, and p.

No arrays, no extra memory.

Memory does not increase with N.

$O(1)$.

