DAA

KURRA . VENKATA GOKUL

CH.SC.U4CSE24121

CSE-B

1.FACTORIAL OF GIVEN NUMBER USING RECURSIVE FU NCTION

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.

In main()

int n → 4bytes

int fact → 4 bytes

Space in main = 8 bytes (constant)

In factorial()

Each call to factorial(n)

creates: int n → 4bytes

Space required for the factorial function is 4 * n bytes. The equation for the Space Complexity is 4*n +8. The order of Space Complexity is n.

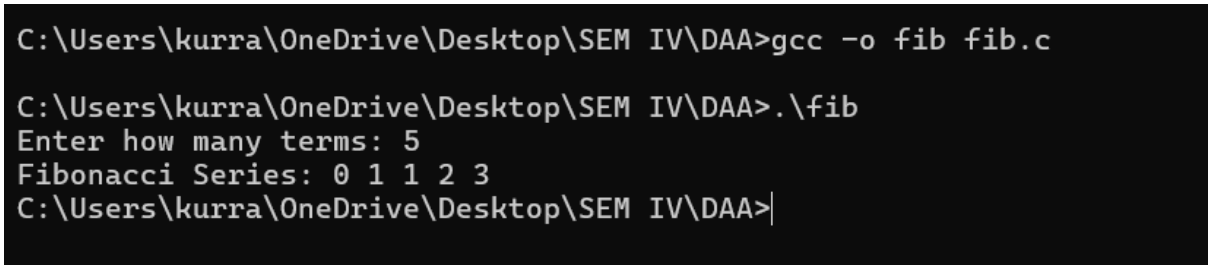
2.FIBONOCCI SERIES

CODE:

File Edit View

```c
#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:



Space complexity :

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

F,f1,fib,loop variable I,input N ---→ each of 4 bytes

Total 20 bytes (constant)

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

O(1) — Constant Space

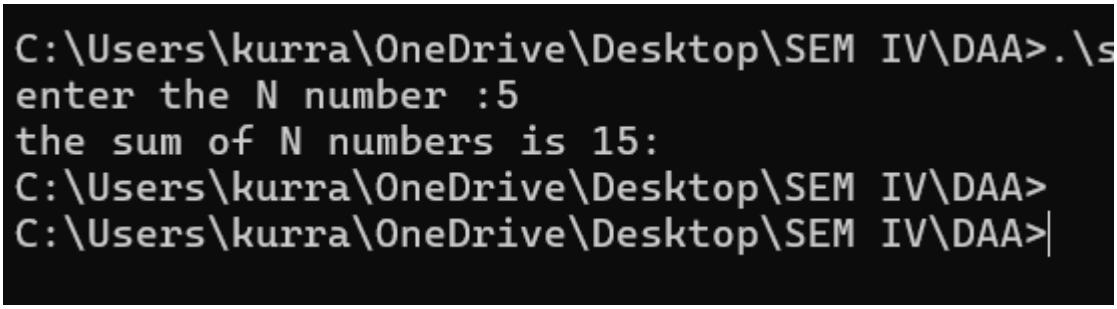
3.SUM OF N NUMBERS

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:



SPACE COMPLEXITY:

In main()

int n → 4bytes

int sum1 → 4 bytes

Total in main() = 8 bytes

In sum()

int sum → 4bytes

int i → 4 bytes

Total in sum() = 8 bytes

Nowthe total bytes required for this program is 16 bytes, which is constant

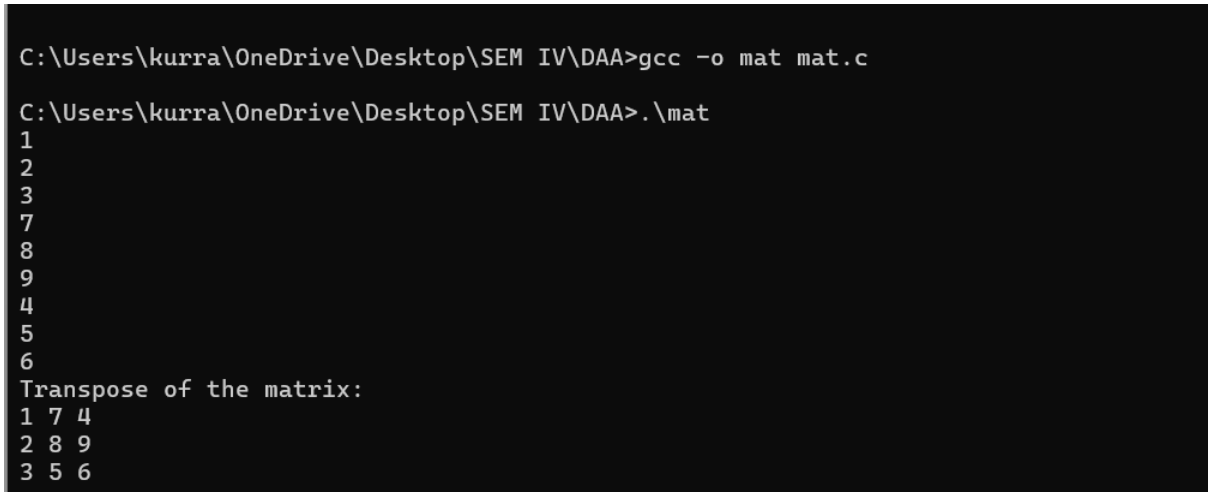
O(1) – CONSTANT SPACE

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:



SPACE COMPLEXITY:

The matrix size is fixed.

Only a few variables are used I,j,arR---→ each of 4 bytes.

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 -→ 4 bytes

total 16 bytes

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 ---→ each of 4 bytes

total 16 bytes

Memory does not increase with N.
O(1).