NAME: D V N Surya Kiran

ROLL NO: CH.SC.U4CSE24116

SUB: DAA- LAB 1 EXP

DATE: 27-11-25

EXP -1 :

CODE :

```c
#include <stdio.h>

int sum(int n)
{
    int s=0;
    for(int i=1;i<=n;i++)
        s+=i;

    return s;
}
int main()
{
    int n;
    printf("Enter a number: ");
    scanf("%d", &n);
    printf("Sum=%d",sum(n));

    return 0;
}
```

OUTPUT :

```
amma@amma13:~$ gcc first.c -o first
amma@amma13:~$ ./first
Enter a number: 7
Sum=28amma@amma13:~$ gcc second.c -o second
```

Space Complexity of the program = O(1)

Because the program uses a fixed amount of memory regardless of the input size n.

EXP -2 :

CODE :

```c
#include<stdio.h>

int main() {
    int n,i;
    int sum=0;

    printf("Enter value of n: ");
    scanf("%d",&n);

    for(i=1;i<=n;i++) {
        sum+=i*i;   // adding square of i
    }
    printf("Sum of squares of first %d natural numbers = %d\n",n,sum);

    return 0;
}
```

OUTPUT :

```
Sum=28amma@amma13:~$ gcc second.c -o second
amma@amma13:~$ ./second
Enter value of n: 8
Sum of squares of first 8 natural numbers = 204
```

Space Complexity = O(1)

Because the program uses a constant amount of memory, no matter what value of n is.

EXP -3 :

CODE :

```c
#include<stdio.h>

int main() {
    int n,i;
    long long sum=0;

    printf("Enter value of n: ");
    scanf("%d",&n);
    for (i=1;i<=n;i++) {
        sum+=(long long)i*i*i;   // adding cube of each number
    }

    printf("Sum of cubes of first %d natural numbers=%lld\n",n,sum);

    return 0;
}
```

OUTPUT :

```
amma@amma13:~$ gcc third.c -o third
amma@amma13:~$ ./third
Enter value of n: 31
Sum of cubes of first 31 natural numbers=246016
```

Space Complexity =0(1)

Because the memory used does **not change** with the value of n.

EXP -4 :

CODE :

```c
#include<stdio.h>

int factorial(int n) {
    if (n==0 || n==1) {
        return 1;
    }
    return n*factorial(n-1);
}

int main() {
    int num;

    printf("Enter a positive integer: ");
    scanf("%d",&num);

    if (num<0) {
        printf("Factorial is not defined for negative numbers.\n");
    } else {
        printf("Factorial of %d=%d\n",num,factorial(num));
    }

    return 0;
}
```

OUTPUT :

```
amma@amma13:~$ gcc fourth.c -o fourth
amma@amma13:~$ ./fourth
Enter a positive integer: 3
Factorial of 3=6
```

Space Complexity =O(n)

Because the recursion stack grows linearly with n.

EXP -5 :

CODE :

```c
#include <stdio.h>

int main() {
    int matrix[3][3], transpose[3][3];

    printf("Enter elements of 3x3 matrix:\n");
    for(int i=0;i<3;i++) {
        for(int j=0;j<3;j++) {
            scanf("%d",&matrix[i][j]);
        }
    }

    for(int i=0;i<3;i++) {
        for(int j=0;j<3;j++) {
            transpose[j][i]=matrix[i][j];
        }
    }

    printf("\nTransposed Matrix:\n");
    for(int i=0;i<3;i++) {
        for(int j=0;j<3;j++) {
            printf("%d ",transpose[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

OUTPUT :

```
amma@amma13:~$ gcc fifth.c -o fifth
amma@amma13:~$ ./fifth
Enter elements of 3x3 matrix:
123 123
1 2
1 3
1 4
1 2 3

Transposed Matrix:
123 2 1
123 1 4
1 3 1
```

Space Complexity = O(1)

Even though you use arrays, their size is **constant (3×3)**, so the memory used does **not grow with input size**.

EXP -6 :

CODE :

```c
#include <stdio.h>

int main() {
    int n,i;
    int a=0,b=1,c;

    printf("Enter the number of terms: ");
    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 :

```
amma@amma13:~$ gcc sixth.c -o sixth
amma@amma13:~$ ./sixth
Enter the number of terms: 4
amma@amma13:~$
```

Space Complexity = O(1)

Memory usage does *not* increase with the number of Fibonacci terms.

---- THE END ----