

NAME : M.SHESHANK

ROLLNO: CH.SC.U4CSE24125

Questions :

1. write a program to find sum of first n natural numbers using user defined functions.
2. write a program to find sum of squares of first n natural numbers.
3. write a program to find sum of cubes of n natural numbers.
4. write a program to find factorial of the given integer using recursion.
5. write a program for transposing 3x3 matrix.
6. write a program to find fibonacci series.

Solutions :

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

```
amma@amma22:~/Documents/125$ gcc -o sum sum.c
amma@amma22:~/Documents/125$ ./sum
enter a number n:
2
```

**Space Complexity:**  $O(1)$  – uses only a fixed number of integer variables (i, sum, n, result) and no additional data structures, so memory usage does not grow with the input size.

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

```
amma@amma22:~/Documents/125$ gcc -o square square.c
amma@amma22:~/Documents/125$ ./square
enter a number n:
2
the sum of 2 natural numbers is 5amma@amma22:~/Documents/1
```

**Space Complexity:  $O(1)$**  – uses only a fixed number of integer variables (n, i, sum) and no additional data structures, so memory usage does not grow with the input size.

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

```
amma@amma22:~/Documents/125$ gcc -o cube cube.c
amma@amma22:~/Documents/125$ ./cube
enter a number n:
2
the cube of 2 natural numbers is 9amma@amma22:~$
```

**Space Complexity:  $O(1)$**  – uses only a fixed number of integer variables (n, i, sum) and no additional data structures, so memory usage does not grow with the input size.

```
#include <stdio.h>
int factorial(int n){
    if (n == 0 || n == 1)
        return 1;
    else
        return n * factorial(n - 1);
}
int main(){
    int num, result;
    printf("Enter a number: ");
    scanf("%d", &num);
    if (num < 0)
        printf("negative.\n");
    else{
        result = factorial(num);
        printf("factorial of %d is %d\n", num, result);
    }
    return 0;
}
```

```
anna@anna22:~/Documents/125$ gcc -o factorial factorial.c
anna@anna22:~/Documents/125$ ./factorial
Enter a number: 3
factorial of 3 is 6
anna@anna22:~/Documents/125$
```

**Space Complexity:  $O(n)$**  – due to recursion, each function call uses stack memory, and for input  $n$ , a total of  $n$  stack frames are created, so memory usage grows linearly with the input size.

```

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

```

```

0 1 1 2 amma@amma22:~/Documents/125$ gcc -o transpose transpose.c
amma@amma22:~/Documents/125$ ./transpose
Enter nums:
1 2 3 4 5 6 7 8 9
transpose of matrix:1 4 7
2 5 8
3 6 9
amma@amma22:~/Documents/125$

```

**Space Complexity:  $O(1)$**  – uses a fixed-size  $3 \times 3$  integer array (`a[3][3]`) and a constant number of loop variables (`i`, `j`), so memory usage does not grow with the input size.

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

```
amma@amma22:~/Documents/125$ gcc -o fibonacci fibonacci.c
amma@amma22:~/Documents/125$ ./fibonacci
Enter number of terms: 4
Fibonacci Series:
0 1 1 2 amma@amma22:~/Documents/125$
```

**Space Complexity:  $O(1)$**  – uses only a fixed number of integer variables (n, a, b, c, i) and no additional data structures, so memory usage does not grow with the input size.