

Design Analysis and Algorithm – Lab Work

Week 1

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

Code:

```
#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 n: ");
    scanf("%d", &n);
    int result = sum(n);
    printf("Sum of first %d natural numbers = %d\n", n, result);
    return 0;
}
```

Output:

Output

```
Enter n: 4
Sum of first 4 natural numbers = 10
```

Space Complexity:

The program uses only a few integers n, i, s

The space complexity is O(1)

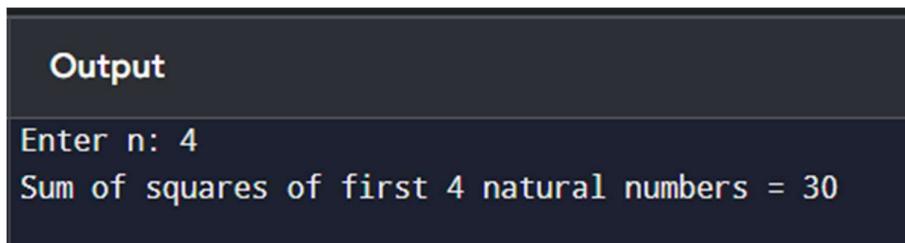
Question 2: Write a program to find sum of squares of the first n natural numbers.

Code:

```
#include <stdio.h>

int main() {
    int n, sum = 0;
    printf("Enter n: ");
    scanf("%d", &n);
    for(int i = 1; i <= n; i++) {
        sum += i * i;
    }
    printf("Sum of squares of first %d natural numbers = %d\n", n, sum);
    return 0;
}
```

Output:



The screenshot shows a terminal window with a dark background and light-colored text. At the top, it says "Output". Below that, the user enters "Enter n: 4" followed by "Sum of squares of first 4 natural numbers = 30".

```
Output
Enter n: 4
Sum of squares of first 4 natural numbers = 30
```

Space Complexity:

The memory used does not change with the value of n.

Space Complexity: O(1)

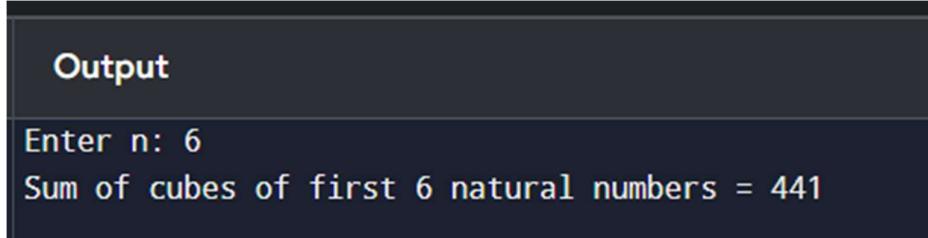
Question 3: Write a program to find sum of cubes of the first n natural numbers.

Code:

```
#include <stdio.h>

int main() {
    int n, sum = 0;
    printf("Enter n: ");
    scanf("%d", &n);
    for(int i = 1; i <= n; i++) {
        sum += i * i * i;
    }
    printf("Sum of cubes of first %d natural numbers = %d\n", n, sum);
    return 0;
}
```

Output:



The image shows a terminal window with a dark background and light-colored text. The title bar says "Output". The window contains the following text:
Enter n: 6
Sum of cubes of first 6 natural numbers = 441

Space Complexity:

The program uses only a few integer

Space Complexity: O(1)

Question 4: Write a program to find the factorial of a given integer using recursion.

Code:

```
#include <stdio.h>

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

int main() {
    int n;
    printf("Enter n: ");
    scanf("%d", &n);
    int result = factorial(n);
    printf("Factorial of %d = %d\n", n, result);
    return 0;
}
```

Output:

Output

```
Enter n: 6
Factorial of 6 = 720
```

Space Complexity:

For an input of n the function calls itself n times, so the program repeats n times.

Space Complexity: O(n)

Question 5: Write a program for transposing a 3 x 3 matrix.

Code:

```
#include <stdio.h>

int main() {
    int a[3][3], t[3][3], i, j;
    printf("Enter elements of 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++){
            t[j][i]=a[i][j];
        }
    }
    printf("Transpose of the matrix:\n");
    for(i=0;i<3;i++){
        for(j=0;j<3;j++){
            printf("%d ",t[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

Output:

Output	
Enter elements of 3x3 matrix: 1 2 3 4 5 6 7 8 9 Transpose of the matrix: 1 4 7 2 5 8 3 6 9	

Space Complexity:

The program uses only two 3x3 matrices with fixed size

Space Complexity: O(1)

Question 6: Write a program to calculate Fibonacci of a number.

Code:

```
#include <stdio.h>

int fibonacci(int n){
    if(n==0) return 0;
    if(n==1) return 1;
    return fibonacci(n-1)+fibonacci(n-2);
}

int main(){
    int n;
    printf("Enter n: ");
    scanf("%d",&n);
    printf("Fibonacci of %d = %d\n",n,fibonacci(n));
    return 0;
}
```

Output:

Output
Enter n: 7 Fibonacci of 7 = 13

Space Complexity:

Since this program uses recursion and each recursive call is stored on the stack

Space Complexity: $O(n)$