DESIGN AND ANALYSIS OF ALGORITHMS
LAB WORKBOOK
WEEK - 1

NAME : Konda Bhavani

ROLL NUMBER : CH.SC.U4CSE24120

CLASS : CSE-B

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

CODE:

```c
#include <stdio.h>
int Sum(int n) {
    int sum = 0;
    for(int i = 1; i <= n; i++) {
        sum += i;
    }
    return sum;
}

int main() {
    int n;
    printf("Enter a number: ");
    scanf("%d", &n);
    printf("Sum of first %d natural numbers is: %d\n", n, Sum(n));

    return 0;
}
```

OUTPUT:

```
Enter a number: 3
Sum of first 3 natural numbers is: 6
```

Justification:

int n → 4 bytes

int i → 4 bytes

int sum → 4 bytes

Inside function Sum():

int sum → 4 bytes

int i → 4 bytes

Total used = 20 bytes, which is a constant.

Space Complexity = O(1).

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

CODE:

```c
#include <stdio.h>
int main(){
    int n;
    int sum=0;
    printf("Enter the number of numbers: ");
    scanf("%d", &n);
    for(int i =n;i>0;i--){
        sum+=i*i;
    }
    printf("The sum of squares of first %d natural numbers is %d\n", n, sum);
    return 0;
}
```

OUTPUT:

```
Enter the number of numbers: 5
The sum of squares of first 5 natural numbers is 55
```

Justification:

int n → 4 bytes

int sum → 4 bytes

int i → 4 bytes

Total used = 12 bytes, which is constant.

Space Complexity = O(1)

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

CODE:

```c
#include <stdio.h>
int main(){
    int n;
    int sum=0;
    printf("Enter the number of numbers: ");
    scanf("%d", &n);
    for(int i =n;i>0;i--){
        sum+=i*i*i;
    }
    printf("The sum of cubes of first %d natural numbers is %d\n", n, sum);
    return 0;
}
```

OUTPUT:

```
Enter the number of numbers: 3
The sum of cubes of first 3 natural numbers is 36
```

Justification:

int n → 4 bytes

int sum → 4 bytes

int i → 4 bytes

Total = 12 bytes (constant).

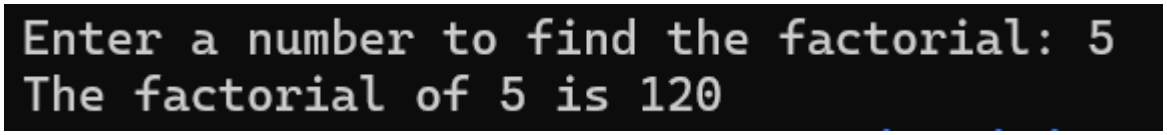
Space Complexity = O(1).

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

CODE:

```c
#include <stdio.h>
int factorial(int n){
    if (n == 1){
        return 1;
    }
    else{
        return n*factorial(n-1);
    }
}
int main() {
    int n, fact;
    printf("Enter a number to find the factorial: ");
    scanf("%d", &n);
    fact = factorial(n);
    printf("The factorial of %d is %d\n",n, fact);
    return 0;
}
```

OUTPUT:



Justification:

In main():

int n → 4 bytes

int fact → 4 bytes

In recursive factorial():

int n → 4 bytes

But each recursive call pushes a new activation record, so space used grows with n.

Total per call = 4 bytes
Total for n calls = 4n bytes

Total = 8+4n bytes

Space Complexity = O(n)

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

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("\nTranspose of the 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:

```
Enter elements of 3x3 matrix:
1 3 5 7 9 1 2 4 6

Transpose of the matrix:
1 7 2
3 9 4
5 1 6
```

Justification:

matrix[3][3] → 9 integers → 9 × 4 = 36 bytes

transpose[3][3] → 9 integers → 36 bytes

int i → 4 bytes

int j → 4 bytes

Total = 36 + 36 + 4 + 4 = 80 bytes (constant)

Space Complexity = O(1)

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

CODE:

```c
#include <stdio.h>
int main(){
  int n,i;
  int a=0, b=1, c;
  printf("Enter the number of Fibonacci Numbers: ");
  scanf("%d" , &n);
  if(n<=0){
    printf("Please enter a positive integer");
    return 0;
  }
  printf("Fibonacci Series: ");
  for (i=1; i<=n;i++){
    if (i==1){
      printf("%d ", a);
      continue;
    }
    if (i==2){
      printf("%d ",b);
      continue;
    }
    c = a+b;
    printf("%d ", c);
    a=b;
    b=c;
  }
  printf("\n");
  return 0;
}
```

OUTPUT:

```
Enter the number of Fibonacci Numbers: 10
Fibonacci Series: 0 1 1 2 3 5 8 13 21 34
```

Justification:

int n → 4 bytes

int i → 4 bytes

int a → 4 bytes

int b → 4 bytes

int c → 4 bytes

Total = 20 bytes (constant).

Space Complexity = O(1)