DESIGN AND ANALYSIS OF ALGORITHMS

LAB WORKBOOK

WEEK - 1

Name: M. charitha Varshini

Roll No: CH.SC.U4CSE24128

Section : CSE-B

1.write program to find sum on n natural numbers using user define function.

Code:

```c
#include<stdio.h>
    int sumOfNaturalNumbers(int n){
    int i,sum=0;
    for(i=1;i<=n;i++){
    sum+=i;
    }
    return sum;
}
int main(){
    int n;
    printf("Enter the no.of natural numbers:");
    scanf("%d",&n);
    int z;
    z=sumOfNaturalNumbers(n);
    printf("%d\n",z);
    return 0;
}
```

Output:

```
21amma@amma25:~$ gcc -o c1 c1.c
amma@amma25:~$ ./c1
Enter the no.of natural numbers:6
21
```

Justification:

In main()

Int n- 4 bytes

Int i- 4 bytes

Int sum- 4 bytes

Total main() = 12 bytes

In sumOfNaturalNumbers()

Int i- 4 bytes

Int sum- 4 bytes

Total in function = 8 bytes

Total memory used:

12 + 8 = 20 bytes (constant)

Space Complexity = O(1)

Because the program uses a fixed amount of space.

2. write a program to find sum of squares of first n natural Numbers

Code:

```c
#include<stdio.h>
int main(){
    int n;
    printf("Enter the  value:");
    scanf("%d",&n);
    int i,sum=0;
    for(i=1;i<=n;i++){
    sum+=i*i;
    }


    printf("%d\n",sum);
    return 0;
}
```

Output:

```
amma@amma25:~$ gcc -o c2 c2.c
amma@amma25:~$ ./c2
Enter the  value:6
91
```

Justification:

n – integer (4 bytes)

i - integer (4 bytes)

sum - integer (4 bytes)

Total – 12 bytes (constant)

Space Complexity = O(1)

Because the memory used does not grow with n.
Only a constant number of variables are used.

3. write a program sum of cubes of first n natural numbers

Code:

```c
#include<stdio.h>
int main(){
    int n;
    printf("Enter the  value:");
    scanf("%d",&n);
    int i,sum=0;
    for(i=1;i<=n;i++){
    sum+=i*i*i;
    }
    printf("%d\n",sum);
    return 0;
}
```

Output:

```
amma@amma25:~$ gcc -o c3 c3.c
amma@amma25:~$ ./c3
Enter the  value:4
100
```

Justification:

n- integer (4 bytes)

i- integer (4 bytes)

sum - integer (4 bytes)

Total= 12 bytes

No arrays ,recursion used

The loop runs n times, but the number of variables does **not** increase with n.

Space Complexity = O(1)

Because the program uses a fixed amount of memory.

4. write a program to find factorial of a given integer using recursion

Code:

```c
#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 number:");
    scanf("%d",&n);
    int x;
    x=fact(n);
    printf("The factorial of %d is %d\n:",n,x);
    return 0;
}
```

Output:

```
amma@amma25:~$ gcc -o c4 c4.c
amma@amma25:~$ ./c4
Enter the number:6
The factorial of 6 is 720
```

Justification:

In main()

Int n- 4 bytes

Int x- 4 bytes

Total = 8 bytes

In fact() -> int n- 4 bytes

There are **n** calls. so n*4 bytes

Main: **8 bytes**
Recursion: **4n bytes**

Total= 4n+8 bytes depends on **n**

Space Complexity = O(n)

5. write a program for transposing a 3*3 matrices

Code:

```c
#include <stdio.h>
int main(){
int A[3][3];
printf("enter matrix elements\n");
for(int i=0;i<3;i++){
    for(int j=0;j<3;j++){
     scanf("%d",&A[i][j]);
     }
     }
     printf("the transpose of matrix is\n");
for(int i=0;i<3;i++){
    for(int j=0;j<3;j++){
     printf("%d ",A[j][i]);
    }
    printf("\n");

     }
     return 0;
     }
```

Output:

```
amma@amma25:~$ gcc -o c6 c6.c
amma@amma25:~$ ./c6
enter matrix elements
3 4 5
6 7 8
2 3 4
the transpose of matrix is
3 6 2
4 7 3
5 8 4
```

Justification:

int A[3][3] – 9*4=36 bytes

int I – 4 bytes

int j – 4 bytes

Total=36+4+4=44 bytes(constant)

Space Complexity = O(1)

The size is **constant and does not depend on input size**, so it is **O(1)**.

6. write a program to find fibonacci series

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: 9
Fibonacci Series: 0 1 1 2 3 5 8 13 21
```

Justification:

Int n- 4 bytes

Int a- 4 bytes

Int b- 4 bytes

Int c-4 bytes

Total=20 bytes

The memory used **does not grow with input n**.

Space Complexity = O(1)