PROGRAM – 02

PART – 2 A

Aim - WAP to Sum array elements.

❖ **Algorithm**

```
Sum (A,n){
S = 0;     ----------------------------------------------------------  (1) unit time
For(i=0 ; i<n ; i++){  -------------------------------------------  (n + 1) unit time
      S = S + A[i];  ---------------------------------------------  (n) unit time
}
Return S; --------------------------------------------------------  (1) unit time
}
```

❖ **Time complexity**

$T(n) = 1 + (n + 1) + n + 1 \quad = \quad 2n + 3$

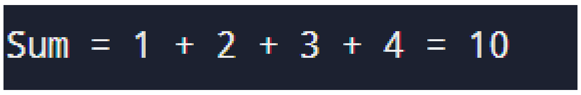
❖ **Space Complexity**

A = n	S(n) = n + 1 + 1 + 1
n = 1	= n + 3
S = 1	= O(n)
i = 1	

❖ **Source Code**

```
#include <stdio.h>
int Sum(int A[], int n) {
   int S = 0;
   for (int i = 0; i < n; i++) {
      S += A[i];
      printf("%d ", A[i]);
      if (i < n - 1) printf("+ ");
   }
   return S;
}
int main() {
   int n;
   printf("Enter the size for Array\n");
   scanf("%d", &n);
   int A[n];
   printf("Enter elements\n");
   for (int i = 0; i < n; i++) {
      scanf("%d", &A[i]);
   }
   printf("Sum = ");
   int result = Sum(A, n);
   printf("= %d\n", result);
   return 0;
}
```

❖ **Output**



```
Sum = 1 + 2 + 3 + 4 = 10
```

PART - 2B

Aim - WAP to add the element of given 2 matrix of N x N.

❖ **Algorithm**

```
Add ( A, B, n ){
For(i=0 ; i<n ; i++){
    For(j=0 ; j<n ; j++){
        C[i,j] = A[i,j] + B[i,j] ;
    }
  }
}
```

❖ **Time Complexity**

$T(n) = (n + 1) + (n(n + 1)) + (n^2) \quad = \quad 2n^2 + 2n + 1$

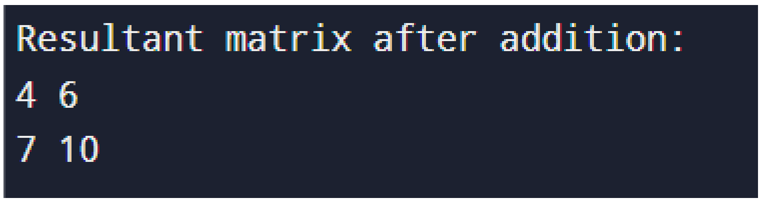
❖ **Space Complexity**

$S(n) = 3n^2 + 3 = O(n^2)$

❖ **Source Code**

```
#include <stdio.h>

void Add(int A[][10], int B[][10], int C[][10], int n) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            C[i][j] = A[i][j] + B[i][j]; // Matrix addition
        }
    }
}
int main() {
    int n;
    printf("Enter the size of the matrix (N x N): ");
    scanf("%d", &n);

    int A[10][10], B[10][10], C[10][10];

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

    printf("Enter elements of second matrix:\n");
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            scanf("%d", &B[i][j]);
    Add(A, B, C, n);
    printf("Resultant matrix after addition:\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++)
            printf("%d ", C[i][j]);
        printf("\n");
    }
    return 0;
}
```

- **Output**



```
Resultant matrix after addition:
4 6
7 10
```

PART - 2C

AIM : WAP to multiply the element of given 2 matrix of N x N.

- **Algorithm**

```
Mul ( A,B,n ){
    For(i=0 ; i<n ; i++){
        For(j=0 ; j<n ; j++){
            C[i,j] = 0;
            For(k=0 ; k<n ; k++){
                C[i,j] = A[i,j] * B[i,j];
            }
        }
    }
}
```

- **Time Compilexity**

$T(n) = (n +1) + (n(n + 1)) + (n^2) + (n(n(n + 1))) + (n^3)$
$= 2n^3 + 3n^2 + 2n + 1$
$= O(n^3)$

- **Space Complexity**

$S(n) = n^2 + n^2 + n^2 + 1 +3$
$= 3n^2$

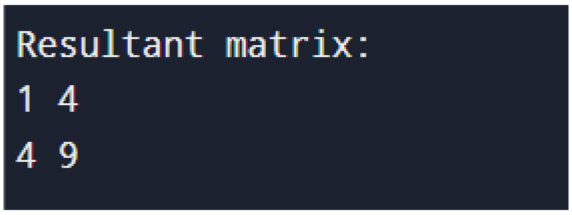
- **Source Code**

```
#include <stdio.h>

void Mul(int A[][10], int B[][10], int C[][10], int n) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
```

Chirag Singh
CSE-23/026

```
                C[i][j] = 0;
                for (int k = 0; k < n; k++) {
                    C[i][j] = A[i][j] * B[i][j]; // Element-wise multiplication
                }
            }
        }
    }
    int main() {
        int n;
        printf("Enter the size of the matrix (N x N): ");
        scanf("%d", &n);

        int A[10][10], B[10][10], C[10][10];

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

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

        Mul(A, B, C, n);

        printf("Resultant matrix:\n");
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++)
                printf("%d ", C[i][j]);
            printf("\n");
        }

        return 0;
    }
```

❖ **Output**



```
Resultant matrix:
1 4
4 9
```