ADD(A,B)

//this function takes 2 NxN type matrixes as parameters

//and it returns a matrix C, that results from the addition of every element

//of matrix A with the element that it corresponds to it from the matrix B

for i<-1 to noOfRows O(n)

for j<- 1 to noOfColumns O(n\*n)

C[i][j]=A[i][j]+B[i][j] O(n\*n)

return C O(1)

//Run Time of the ADD function is O(2n^2+n+1)=O(n^2)

SUB(A,B)

//this function takes 2 NxN type matrixes as parameters

//and it returns a matrix C, that results from the subtraction of every element

//of matrix A with the element that it corresponds to it from the matrix B

for i<-1 to noOfRows O(n)

for j<- 1 to noOfColumns O(n\*n)

C[i][j]=A[i][j]-B[i][j] O(n\*n)

return C O(1)

//Run Time of the SUB function is O(2n^2+n+1)=O(n^2)

MULTIPLY\_WITH\_SCALAR(A,k)

//this function takes a NxN matrix A and a constant k as parameters

//it returns a matrix B, that results from multiplying

// every element of the matrix with the constant

for i<-1 to noOfRows O(n)

for j<- 1 to noOfColumns O(n\*n)

B[i][j]=A[i][j]\*k O(n\*n)

return C O(1)

//Run Time of the MULTIPLY\_WITH\_SCALAR function is O(2n^2+n+1)=O(n^2)

MULTIPLY(A,B)

//this function take 2 NxN matrixes as parameters

//and it return a matrix C, that results from the multiplication

//of the 2 matrixes

for i<-1 to noOfRows O(n)

for j<- 1 to noOfColumns O(n\*n)

s<-0

for k<-1 to nrOfColumns/Rows O(n\*n\*n)

s<-s+A[i][k]\*B[k][j] O(n\*n\*n)

C[i][j]<-s O(n\*n\*n)

return c O(1)

//Run Time of the MULTIPLY function is O(3\*n^3+n^2+n+1)=O(n^3)

A<-SUB(MULTIPLY(B,C),MULTIPLY\_WITH\_SCALAR(ADD(B,C),2))

Total Run Time: O(n^3+3n^2)=O(n^3)