PSUEDO CODE

Name: Nithin S

Roll number: 221IT085

BEGIN

Function allocateMatrix

// Function to dynamically allocate matrix in heap

INPUT : No. Of Rows and Columns.

OUTPUT : Pointer to an array of Row Pointers.

matrix = Pointer to an Array of Row Pointers (Double Pointer to a matrix)

// Allocate memory for the array of row pointers

matrix = Allocate memory for Array of Integer Pointers with size equal to rows

if matrix is NULL *// If Allocation Fails*

Return NULL

// Loop to allocate memory for each row pointer

For i <- 0 to rows do

matrix[i] = Allocate memory for Array of Integers with size equal to columns

if matrix[i] is NULL *// If Allocation Fails*

Return NULL

Return matrix

Function multiplyMatrices

```
// Function to Recursively multiply two Matrices
   INPUT    :  Doube Pointer to Matrix A,Matrix B and Resultant Matrix ,
               No. Of Rows rA and Columns cA of Matrix A,
               No. Of Rows rB and Columns cB of Matrix B
   OUTPUT :  Double Pointer to Resultant Matrix

   static Interger variables i = 0, j = 0, k = 0

   If i >= rA  Then
       Return NULL
   Else If i >= rA Then
        If j<cB
              if k<cA
                    R[i][j] += A[i][k] * B[k][j]
                    k++
                    Call multiplyMatrices(A,B,R,rA,cA,rB,cB)
              k=0
              j++
              Call multiplyMatrices(A,B,R,rA,cA,rB,cB)
        j=0
        i++
        Call multiplyMatrices(A,B,R,rA,cA,rB,cB)
Return R
```

Function printMatrix

// Function to print the 2D Array

INPUT : Double Pointer to 2D Dynamic Array , No. Of Rows and Columns

OUTPUT: Void

For i<- 0 to rows

For j<- 0 to columns

Print matrix[i][j]

Print newline

Function freeMatrix

// Function to free dynamically allocated memory in heap

INPUT : Double pointer to 2D Dynamic Array and No. Of Rows

OUTPUT : Void

For i<- 0 to rows

Free memory allocated for matrix[i]

Free memory allocated for matrix

Set matrix to NULL

Function fillRandomValues

// Function to fill the 2D Dynamic matrix with random values

INPUT: Double Pointer to matrix , No. Of Rows and Columns

OUTPUT : Void

Function fillRandomValues(matrix, rows, cols, lb, ub)

For i <- 0 to rows

For j <- 0 to cols

// Generate random values between 1 and 100, inclusive

value = Generate random integer between 1 and 100

matrix[i][j] = value

Function Main

INPUT : None

OUTPUT : 0

A = Double Pointer to Matrix A

rA = No. Of Rows of Matrix A

cA = No. Of Columns of Matrix A

B = Double Pointer to Matrix B

rB= No. Of Rows of Matrix B

cB = No. Of Columns of Matrix B

R = Double Pointer to Resultant Matrix

Do

Print "Enter the dimensions (rows columns) of the first matrix separated by spaces: "

Read rA, cA

Print "Enter the dimensions (rows columns) of the second matrix separated by spaces: "

Read rB, cB

if (rA==0 or rB==0 or cA==0 or cB==0)

Print “Invalid Dimensions! Please enter a valid number.”

Else If (rA<10 or rB<10 or cA<10 or cB<10)

Print “Minimum row and column dimension is 10.Please enter valid dimension”

While (rA<10 or rB<10 or cA<10 or cB<10)

If cA != rB Then

Print "Matrix Multiplication is not possible!"

Return 0

A = allocateMatrix(rA, cA)

```
B = allocateMatrix(rB, cB)
R = allocateMatrix(rA, cB)

Seed random number generator with current time
fillRandomValues(A, rA, cA)
Seed random number generator with a new seed
fillRandomValues(B, rB, cB)

Print "Matrix 1:"
printMatrix(A, rA, cA)
Print "Matrix 2:"
printMatrix(B, rB, cB)

multiplyMatrices(A, B, R, rA, cA, rB, cB)
Print "The Result of multiplication:"
printMatrix(R, rA, cB)

freeMatrix(A, rA)
freeMatrix(B, rB)
freeMatrix(R, rA)

Return 0
```

STOP