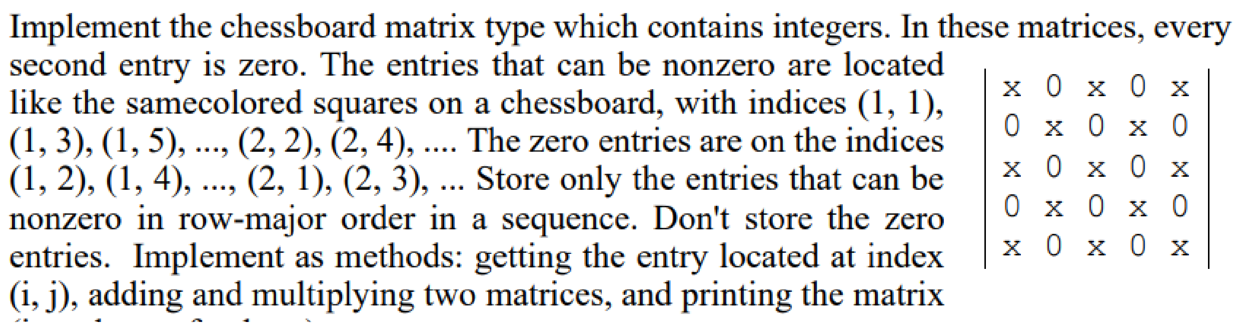
Name: Mamatova Bermet
Date: March 22th, 2024
Neptune code: BTA5BP
Group:6
Email: bta5bp@inf.elte.hu

Assignment 1/Task 1

Problem task:


Implement the chessboard matrix type which contains integers. In these matrices, every second entry is zero. The entries that can be nonzero are located like the samecolored squares on a chessboard, with indices (1, 1), (1, 3), (1, 5), ..., (2, 2), (2, 4), ... The zero entries are on the indices (1, 2), (1, 4), ..., (2, 1), (2, 3), ... Store only the entries that can be nonzero in row-major order in a sequence. Don't store the zero entries. Implement as methods: getting the entry located at index (i, j), adding and multiplying two matrices, and printing the matrix

$$\begin{vmatrix} x & 0 & x & 0 & x \\ 0 & x & 0 & x & 0 \\ x & 0 & x & 0 & x \\ 0 & x & 0 & x & 0 \\ x & 0 & x & 0 & x \end{vmatrix}$$

Set of values

ChessMatrix(n) = { $\forall$ $\mathbb{Z}^{szie \times size}$ i[1..size], j[1..size]: (i+j) mod 2 ≠ 0 → a[i,j]=0 }

Operations

1. *Getting an entry*

   Getting the index of the element of the ith row and jth column (i[1..size], j[1..size]) in the list that represents the matrix. The matrix's rows and columns are the same.
   Formally :

$$\begin{array}{lccccccc} A = & \text{Matrix(size)} & \times & \mathbb{Z} & \times & \mathbb{Z} & \times & \mathbb{Z} \\ & m & & i & & j & & e \end{array}$$

Pre = (m = m' ∧ i = i' ∧ j = j' ∧ i ∈ [1..size], j∈ [1..size])
Post = (Pre∧ e = a[i, j])

2. *Summation*
   The sum of two matrices: third:=first+second. The matrices have the same sizes.

   Formally:

$$\begin{array}{lccccc} A = & \text{Matrix(size)} & \times & \text{Matrix (size)} & \times & \text{Matrix(size)} \\ & A & & B & & sum \end{array}$$

Pre = (a = a' ∧ b = b')

Post = (Pre $\wedge$ $\forall$ i $\in$ [1..size], j $\in$[1..size]: sum[i, j] = a[i,j] + b[i,j])

For chessboard matrices :

$\forall$ i $\in$ [1..size], j $\in$[1..size]: sum[i, j] = a[i,j] + b[i,j] and $\forall$ i $\in$ [1..size], j $\in$[1..size]: (i+j) mod $\neq$ 0, then sum [i, j] = 0

3. *Multiplication*

Multiplication of two matrices prod: = a * b . By the rule, the a's number of columns has to be equal to the b's number of rows

Formally: A = Matrix(p,q) x Matrix(q,r) x Matrix(p,r)

A B prod

Pre = (a = a' $\wedge$ b =b')

Post = (Pre $\wedge$ $\forall$ i $\in$ [1..size], j $\in$[1..size]: prod[i,j] =

$=\Sigma_{k = [1..size]} a[i, k] * b[k, j]$

For chessboard matrices :

$\forall$ i $\in$ [1..size], j $\in$[1..size]: $\Sigma_{k = [1..size]}$ a[i, k] * b[k, j] and $\forall$ i $\in$ [1..size], j $\in$[1..size] (i+j) mod 2 $\neq$ 0 , then prod[i,j] = 0

Representation

Only the elements for which the sum of indexes is even have to be strored.

$$a = \begin{array}{ccccc} a_{11} & 0 & a_{13} & \dots & 0 \\ 0 & a_{22} & 0 & \dots & a_{2n} \\ a_{31} & 0 & a_{33} & \dots & 0 \\ & \dots & & & \dots \\ 0 & a_{m2} & 0 & \dots & a_{mn} \end{array} \quad \leftrightarrow \quad list = < a_{11} a_{13} \dots a_{22} \dots a_{mn} >$$

(in case of even m and n)

Only a one-dimension array (list) is needed, with the help of which any entry of the diagonal matrix can be get:

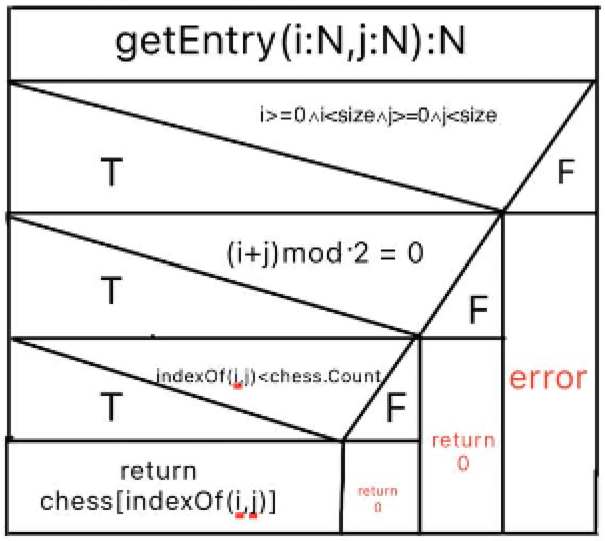
$$a[i,j] = \begin{cases} list[(i*n + j)/2] \ if \ (i+j) \ mod \ 2 = 0, \\ 0, \ if \ (i+j) \ mod \ 2 \neq 0 \end{cases}$$

Implementation

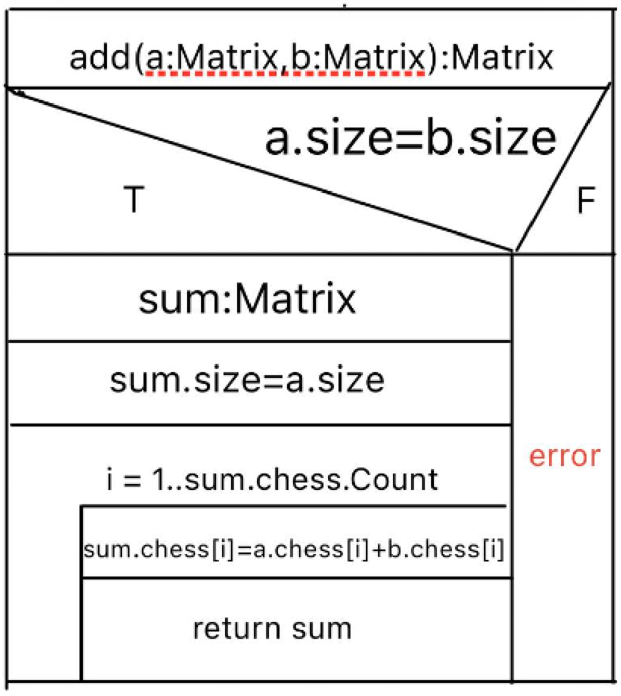
1. **Getting an index in the list**

indexOf(i:N,j:N):N
return (i*size+j)/2

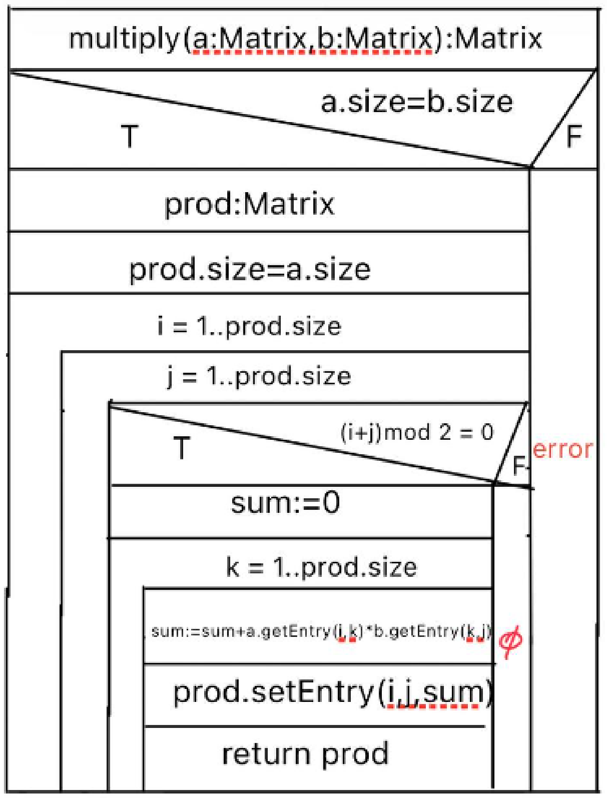
2. **Getting an entry**
   Getting the entry of the ith column and jth row (i[1..size], j[1..szie]) e:=a[i,j] where the matrix is represented by list,1≤ i≤ size, 1≤ j≤ size, can be implemented as:



3. **Sum**
   The sum of matrices a and b goes to matrix sum, where all of the lists of the matrices have to have the same size.



4. **Product**
The product of matrices aand b goes to matrix prod, where the a matrix's number of columns has to be equal to b matrix's number of rows.



Testing

Testing based on the code:

1. TestAdd: This test checks the add method of the Matrix class. It creates two Matrix objects, sets some entries, adds them together, and then checks that the entries in the resulting Matrix are as expected.
2. TestMultiply: This test is checking the multiply method of the Matrix class. It creates two Matrix objects, sets some entries, multiplies them together, and then checks that the entries in the resulting Matrix are as expected.
3. TestGetElem: This test is checking the getEntry and setEntry methods of the Matrix class. It creates a Matrix object, sets some entries, retrieves them using getEntry, and then checks that the retrieved values are as expected.
4. TestIndexOutOfRangeException: This test is checking that the getEntry method of the Matrix class throws an IndexOutOfRangeException when it's asked to retrieve an entry at an index that's out of range.
5. TestNoSizeMatchException: This test checks that the add method of the Matrix class throws a NoSizeMatchException when it's asked to add two Matrix objects of different sizes.