Assignment_ Chessboard Matrix

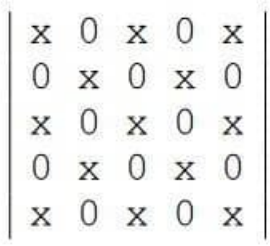
Name: Rahim Adnan

Practice group: 08

Neptun code: gw9k9t

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 (in a shape of m by n).



Chessboard matrix type

Set of values

*C*matrix(*cb*)= { $a \in \mathbb{Z}^{cb \times cb}$ | i,j ∈ [1..m],j ∈ [i..n], i!= j }

Operations

1. Getting an entry

Getting the entry of the *i*th column and *j*th row (*i,j*∈[*1..m*],[1..n]):
list:=*a*[*i,j*].

A : *CB*(*cb*) × ℤ × ℤ × ℤ
Pre = (*a=a'* ⋀ *i=i'* ⋀ *j=j'* ⋀ *i*,j∈[*1..m*],[1..n])
Post = (*Pre* ⋀ *list=a*[*i,j*])

2. Setting an entry

Setting the entry of the *i*th column and *jth* row (*i*,j∈[*1..m*],[1..n): *a*[*i,j*]:=*num*. Entries for which (*i+j) mod 2* ≠ 0 cannot be modified.

A : *CB*(*cb*) × ℤ × ℤ × ℤ
Pre = (*num=num'* ⋀ *a=a'* ⋀ *i=i'* ⋀ *j=j'*⋀ *i,j*∈[*1..m*],[1..n] ⋀ (*i+j) mod 2* != 0)
Post = (*num=num'* ⋀ *i=i'* ⋀ *j=j'*⋀ *list[a]= num*)

3. Sum

Sum of two matrices: *c.list[i]: =this.list[i]+cb*.list[i] The matrices have the same sizes.

$A = CB(cb) \times CB(cb) \times CB(cb)$
Pre = ($cb=cb' \wedge i=i'$)
Post = (Pre $\wedge \forall i,j \in [1..m],[1..n]$: $c[i] = list[i]$ + cb.list[i])

4. Multiplication

Multiplication of two matrices: c*:=a*b*. The first matrix's number of columns must be equal to the second matrix's number of rows.

$A = CB(cb) \times CB(cb) \times CB(cb)$
Pre = ($sum=sum' \wedge i=i' \wedge$ i+k mod 2 !=0|| k+j mod 2!=0)
Post = (*Pre* $\wedge \forall i,j \in [1..m],[1..n]$: $sum=\sum_{k=1..n} this.getElem(i,k) * cb.getElem(k,j))$

Representation

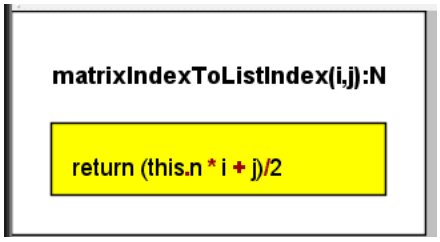
Only the elements for which the sum of indexes is even having to be stored.

$$\begin{matrix} a_{11} & 0 & a_{13} & \dots & 0 \\ 0 & a_{22} & 0 & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & a_{m2} & 0 & \dots & a_{mn} \end{matrix} \qquad list=\langle a_{11},a_{13},\dots,a_{22},\dots,a_{mn}\rangle$$

Implementation

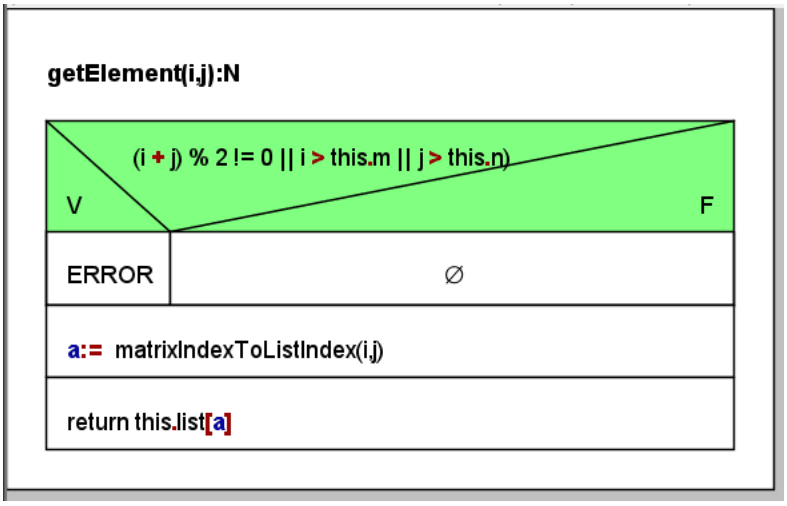
a. MatrixIndexToListIndex():

This method is used to get the Index of the Matrix to set in the ChessboardMatrix, where n is the column of the matrix and i , j represents the index of the matrix.



0. Getting an Entry

Getting the entry of the ith column and jth row (i,j ∈ [1..m] ,[1..n])list:= [i,j]



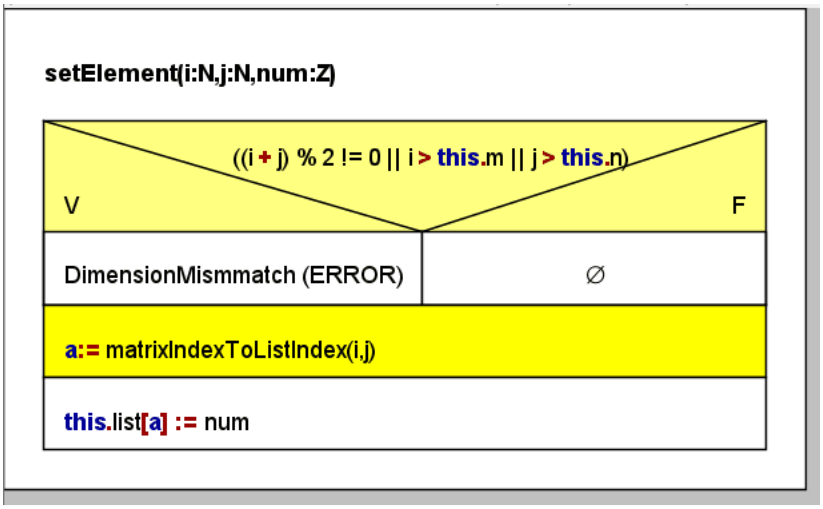
2. Setting an entry

The SetElement method is used to set the value of an element at a specified row and column index in the matrix. It takes three parameters:

i: The row index of the element.

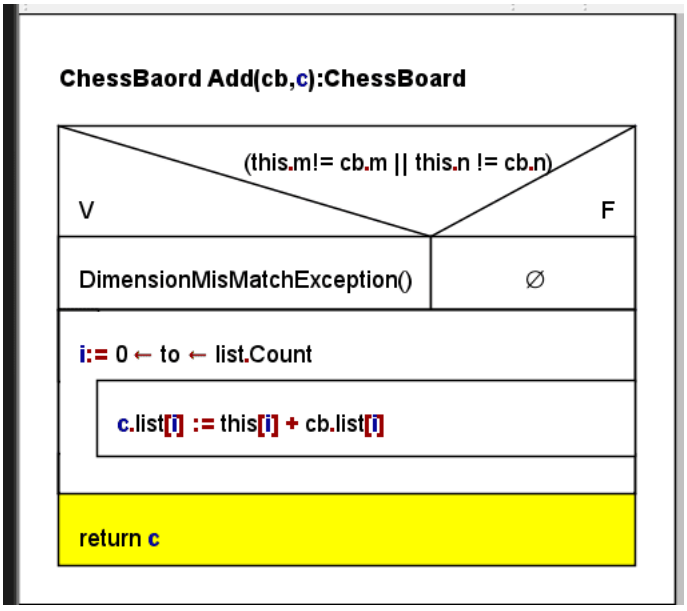
j: The column index of the element.

num: The value to be set at the specified position.



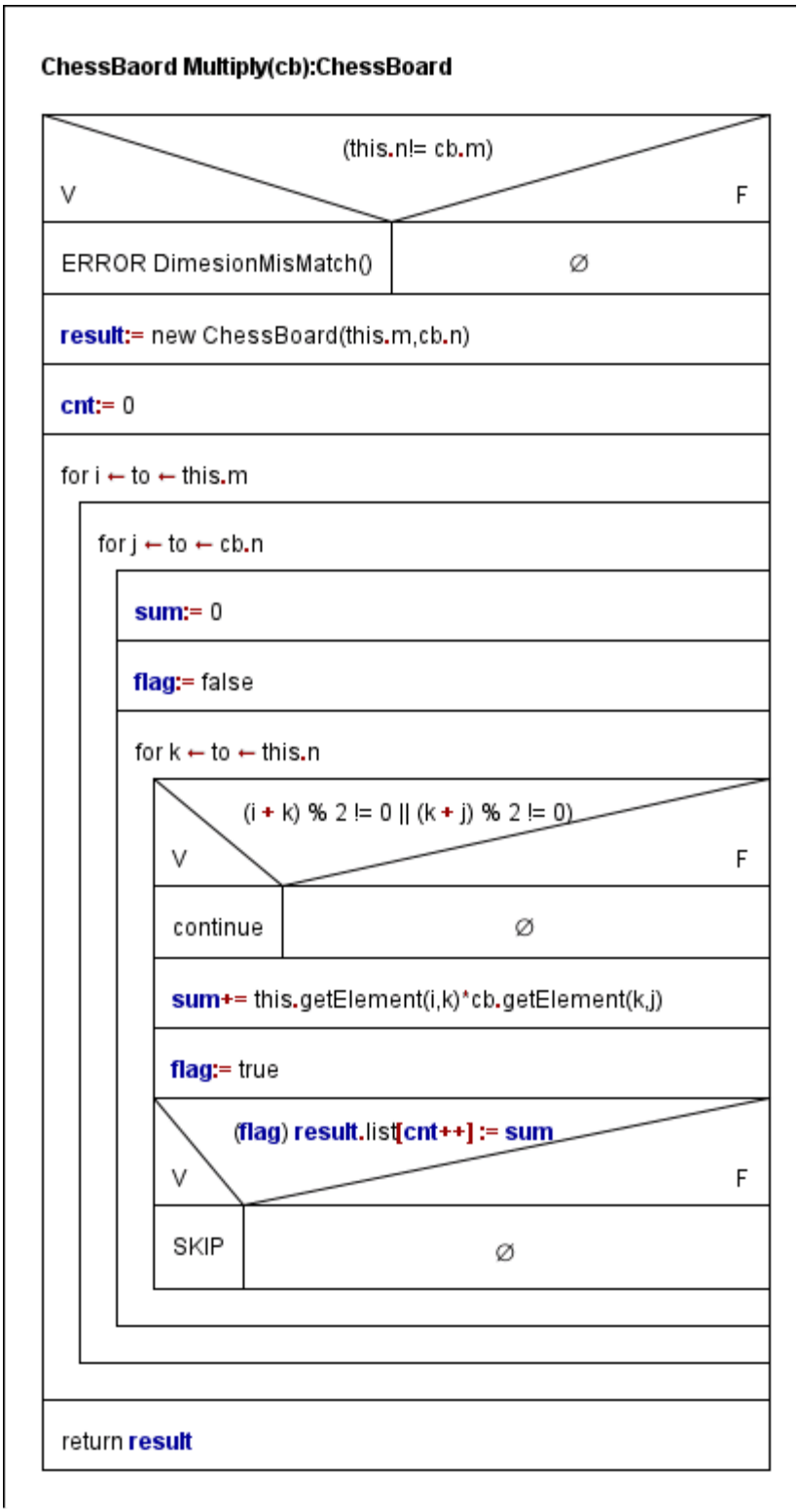
3. Sum

This method in the ChessBoard class adss corresponding elements from two matrices and returns a new ChessBoard object containing the result of the addition.



4. Matrix Multiplication

The product of matrix a and b (represennted by Chessboard matrices matrix annd cb2) goes to the chessboard matrix result, where a's column and b's rows are the same.



Testing

Testing based on the code: (Black Box Testing)

1. Creating , printing , and checking matrices of different size.

a) Creating a 3 x 3 – size matrix

b) Printing a 3x3 -size matrix

c) Reading the printed matrix and checking whether it’s the same one or not

2) Creating and getting an entry

a) creating a 4 x4 – size matrix

b) getting and entry by indexing the row and column number

3) Sum of two matrices , command list:= a[i] + list[i]

a) With the matrix of same size :

b) with matrices of different size

4) Multiplication of two matrices, (list:= sum[i] * sum[j,k])

a)With matrices where column number of the first matrix is equal to the row number of the second matrix.

b) with matrices where column number of the first matrix doesn’t equal to the row number of the second matrix.

Testing based on the Code(White Box Testing) 1>

Generating and catching Exception.