Hussain Izhar, **1. assignment1/1.task** 07th March, 2021
L9EIKJ
l9eikj@inf.elte.hu
Group 3

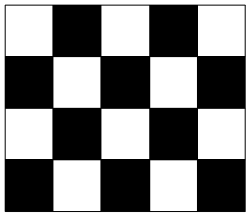
Task:

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).

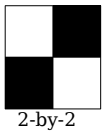
Idea:

Chess type matrices with one non-zero and adjecent two zero squres are of the size 2-by-2, 4-by-4, 6-by-6, ..., n-by-n (where n is even). This implies that chess type matrices even squared matrices. Some of the representations are given below:

Correct Cases:



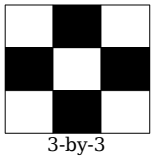
4-by-4



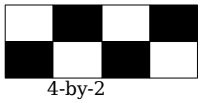
2-by-2

Incorrect Cases:

1-by-1



3-by-3



4-by-2

Reasoning:

When we analyze an 8-by-8 chess matrix, which is a standard board we must agree on the following points:
1. All the possible combination of chess matrices must be of even sizes.
2. If a row of any chess matrix is started from a non-zero digit (say black colour), the same row should terminate on the alternative zero digit (white colour). We know that odd-by-odd matrices disobey the rule. So, we must exclude odd-by-odd matrices.

By above reasoning, we come to a conculsion that chess type matrices are of the sizes n = 2, 4, 6, 8, 10, and so on. One question that comes to mind is what would be the possible NON-ZERO elements (or number of white squares) that can be stored in these matrices? Well, if we work them out by hand, we come to the following conclusion:

Size (N)	No. of non-zero elements elements
2	2
4	8
6	18
8	32
10	50
...	...
N	$2 * N^2$

Set Of Values:

$$\text{ChessMatrix(n)} = \{\ a \in \mathbb{Z}^{n\ x\ n} \mid \forall\ i,j \in [1..n] : i + j\ \%\ 2 \neq 0 \rightarrow a[i, j] = 0\}$$

Operations:

1. Getting and Entry:

Getting an entry at *i*th row and *j*th (i,j ∈ [1..n]): e:=a[i,j])

$$A : \text{ChessMatrix(n)} \times \mathbb{Z} \times \mathbb{Z} \times \mathbb{Z}$$

a i j e

$$\text{Pre} = (\ a=a' \wedge i=i' \wedge j=j' \wedge i,j \in [1..n]\ )$$

$$\text{Post} = (\ \text{Pre} \wedge e=a[i,j]\ )$$

NOTE: This operation needs any action only if $i + j\ \%\ 2 = 0$, otherwise the $e = 0$;

2. Sum:

Adding two matrices c:=a+b, provided that the size of the matrices are same

Formally:

$$A = \text{ChessMatrix(n)} \times \text{ChessMatrix(n)} \times \text{ChessMatrix(n)}$$

a b c

$$\text{Pre} = (\ a=a' \wedge b=b')$$

$$\text{Post} = (\forall\ i, j \in [1..n]:\ \ i + j\ \%\ 2 = 0 \rightarrow c[i,j]:= a[i,j] + b[i,j]\ \ \text{and}\ \forall\ i, j \in [1..n]: i + j\ \%\ 2 \neq 0 \rightarrow c[i, j] = 0)$$

3. Multiply:

Multiplication of two matrices: c:=a*b. The matricess must have the same size. Formally:

$$A = \text{ChessMatrix}(n) \times \text{ChessMatrix}(n) \times \text{ChessMatrix}(n)$$

$$Pre = (\ a{=}a' \frown b{=}b')$$

$$Post = (\ Pre\ \ \forall\ i, j \in [1..n]: c[i,j] = \Sigma_{k=1..n}\ a[i,k] * b[k,j])$$

Representation:

$$a := \begin{matrix} a_{11} & 0 & a_{13} & 0 & a_{15} & \dots & 0 \\ 0 & a_{22} & 0 & a_{24} & 0 & \dots & a_{2n} \\ a_{31} & 0 & a_{33} & 0 & a_{35} & \dots & 0 \\ 0 & a_{42} & 0 & a_{44} & 0 & \dots & a_{4n} \\ a_{51} & 0 & a_{53} & 0 & a_{55} & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & a_{n2} & 0 & a_{n4} & 0 & \dots & a_{nn} \end{matrix} \quad \longleftrightarrow \quad v := \forall\ i, j \in [1..n]: <a_{i,j}> \text{ where } i + j\ \%\ 2 = 0$$

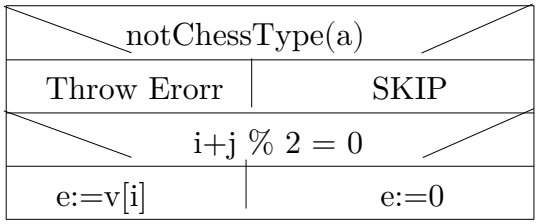
We use only one dimensional vector v to represent the ChessMatirx.

$$a[i, j] = \{v[i] \ \text{ if } i{+}j\ \%\ 2 = 0 \text{ and } 0 \text{ if } i + j\ \%\ 2 \neq 0\}$$

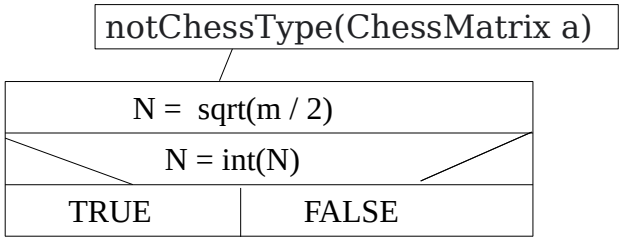
Implementation:

1. Getting an Entry:

Getting an Getting the entry of the ith column and jth row $(i,j{\in}[1..n])$ $e{:=}a[i,j]$ where the matrix is represented by $v,\ 0 \leq i \leq n\text{-}1,$ and n stands for the size of the matrix can be implemented as:

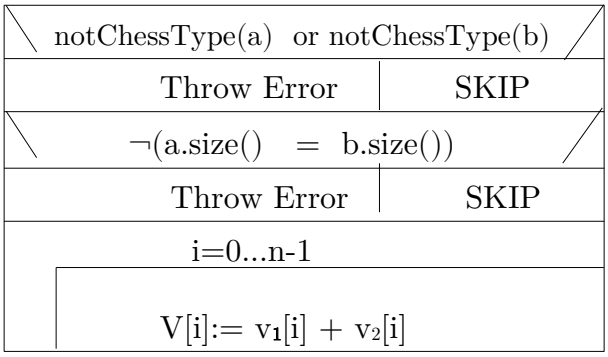


Let the number of non-zero elements inside the ChessMatrix is represented by m, sqrt(n) is defined to be the sqaure root of n, int(n) gives the integer part of a floating point number, and TRUE and FALSE are the logical return values.



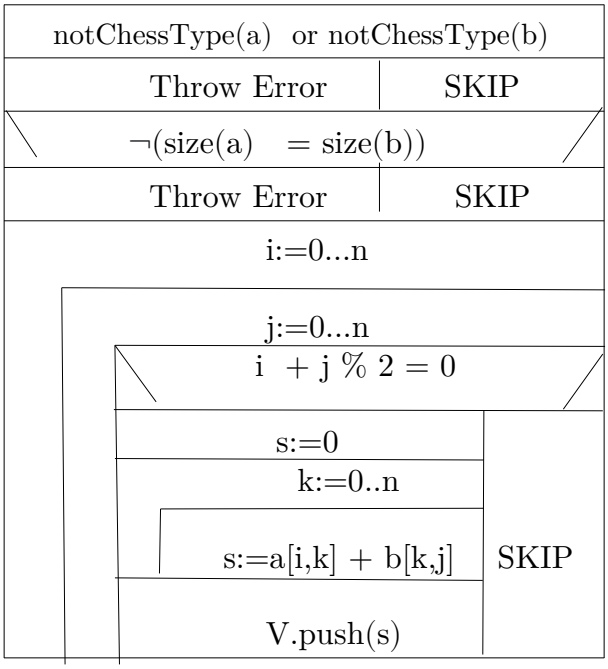
2. Sum:

The sum of matrices a and b (represented by vectors v_1 and v_2) goes to matrix c (represented by array V), where all of the vectors have to have the same size.



3. Multiply:

The product of matrices a and b goes to matrix c (represented by vector V, intiallay empty), where all of the vectors have to have the same size. Let the size of the matrix a, b, and c be n, where size(a) returns the size of a given matrix, and push(x) puts an integer element to the end of the vector which represent the given matrix.



CODE:

You can find the code by opening ChessMatrix.cpp, under ChessBoard/src.

TESTING:

Test Case 1**:**

FILE CONSTRUCTOR: must throw an error on wrong input file.

Test case 2:

VECTOR CONSTRUCTOR: must throw an error on wrong input data in the vector

Test case 3:

SIZE CONSTRUCTOR: must throw an error on on wrong input size.

Test Case 4:

NEGATIVE INDEXES: must throw an error if the given indexes to access an element are negative.

Test Case 4:

INDEX OUT OF BOUND: must throw an error if the given indexes to acces an element are out of bound (i.e., greater than the size of the matrix)

Test Case 5:

CORRECT INDEXES: if provided with corrent indexes (positive and smaller than the size of the matrix), the function must return the expected value inside the matrix.

Test Case 6:

SUM OF TWO DIFFERENT SIZE MATRICES: must throw an error if the given matrices are not of equal sizes.

Test Case 7:

SUM OF TWO NON-CHESS TYPE MATRICES: must throw an error if any of the given matrices is not a chess type matrix

Test Case 8:

SUM OF TWO CORRECT MATRICES: must return the expected sum for two randomly selected chess-type matrices.

Test Case 9:

MATHEMATICAL LAWS OVER ADDITION: must obey all the mathematical law over additions, namely: commutativity, associativity, neutral element (which says 0 + m = m, where m is a chess type matrix).

Test Case 10:

MULTIPLICATION OF TWO DIFFERENT SIZE MATRICES: must throw an error if the given matrices are not of equal sizes.

Test Case 11:

MULTIPLICATION OF TWO NON-CHESS TYPE MATRICES: must throw an error if any of the given matrices is not chess type matrix

Test Case12:

MULTIPLICATION OF CORRECT MATRICES: must return the excepted product for two randomly selected chess type matrices.

Test Case 13:

MATHEMATICAL LAWS OVER MULTIPLICATION: must obey all the laws of mulitplication of matrices, namely, commutativity, associativity, neutral element (which says 0* a= 0, where a is a chess type matrix), and identity matrix multiplication (which says I * m = m, where I is the identity matrix and m is the chess type matrix.)