

```
% CONTENTS.m
% msqr - Magic Square Toolbox
%
%
% Generation of Magic Squares
% msqr_Narayana - Magic Square of Odd Order by Narayana-de la Loubre Method (Chapter 2)
% msqr_Bachet - Magic Square of Odd Order by Bachet's Method (Chapter 3)
% msqr_diagonal - Magic Square of Odd Order by Diagonal Method (Chapter 4)
%
% msqr_dblEven1 - Magic Square of Doubly Even Order I (Chapter 5)
% msqr_dblEven1b - A one-line function implementation of Magic Square of Doubly Even Order I (Chapter 5)
% msqr_dblEven2 - Magic Square of Doubly Even Order II (Chapter 6)
% msqr_Kothari - Magic Square of Doubly Even Order III by using Prof LS Kotharis's method (Chapter 7)
%
% msqr_Kaumudi - Magic Square of Singly Even Order by Narayana's method in Ganita Kaumudi (Chapter 8)
% msqr_Strachey - Magic Square of Singly Even Order by Ralph Strachey's method (Chapter 9)
% msqr_ConwayLUX - Magic Square of Singly Even Order by Conway's LUX method (Chapter 10)
%
%
% Verifying if a Given Square is a Magic Square
% msqr_verify - Checks a given square to see if it is a Magic Square and prints a message
%
%
% Demos
% msqr_demo - commands illustrating the use of all the above functions
%
```

```

%FILE msqr_narayana
function finalSqr = msqr_narayana(n)
% MSQR_NARAYANA : MAGIC SQUARES OF ODD ORDER
% This function generates a magic square of odd order by the Narayana - de la Loubre method
% (Chapter 2: Magic Squares of Odd Order-1)
%
% INPUT: n - the order of the magic square
% OUTPUT : finalSqr -the desired Magic Square
% USAGE sqr = msqr_narayana(n)
% EXAMPLE sqr = msqr_narayana(3);
if mod(n,2)~= 1
    error('The input for the order of the Magic Square is not an Odd Number.');
```

end

```

finalSqr = zeros(n);
middle = 1+(n-1)/2;
% STEP 1: Take an empty square and place the number 1 in the middle cell of the top row.
% STEP 2: Whenever you are in the top row move to the bottom cell of the next column and place
%         the next number there
% STEP 3: Move diagonally upwards from left to right one cell at
%         a time filling up the number in increasing order, keep doing so till this is possible
% STEP 4: When you reach the extreme right column move to the row immediately above and go to
%         the first cell in that row and fill it up with the next number,
%         then apply step 3 as far as possible.
% STEP 5: When you reach a cell that is already occupied move one cell down
%         in the same column and fill it with the next number

%STEP 1
currentRow = 1;
currentColumn = middle;
finalSqr(currentRow,currentColumn) = 1;

k = 2;
while k <= n^2

    %APPLY STEP 2
    if currentRow==1 & currentColumn<n
        currentRow = n;
        currentColumn = currentColumn+1 ;
        finalSqr(currentRow,currentColumn) = k ;
        k = k+1;

        continue;
    end
    % APPLY STEP 3
    if currentRow-1>=1 & currentColumn+1<= n
        currentRow = currentRow-1 ;
        currentColumn = currentColumn+1 ;
        if finalSqr(currentRow,currentColumn) == 0
            finalSqr(currentRow,currentColumn) = k ;
            k = k+1;
        else
            %UNDO THE DIAGONALLY UP MOVE and APPLY STEP 5
            currentRow = currentRow + 1 ;
            currentColumn = currentColumn-1 ;
            %COME DOWN ONE ROW
            currentRow = currentRow +1;
            finalSqr(currentRow,currentColumn) = k ;
            k = k+1;
        end
    end
    continue;
end

% APPLY STEP 4
if currentColumn == n & currentRow > 1
    currentRow = currentRow-1;
    currentColumn = 1 ;
    finalSqr(currentRow,currentColumn) = k ;
    k = k+1;
end

```

```
    continue;
end
if currentColumn == n & currentRow == 1
    currentRow = currentRow + 1;
    finalSqr(currentRow,currentColumn) = k ;
    k = k+1;
    continue;
end
```

```
end
disp(num2str(finalSqr))
%%%%%%%%%
```

```

%FILE : msqr_Bachet
function sqr = msqr_Bachet(n)
% MSQR_BACHET : MAGIC SQUARES OF ODD ORDER
% This function generates a magic square of odd order by the method due to Bachet de Meziric
% (Chapter 3: Magic Squares of Odd Order-II)
%
% INPUT: n - the order of the magic square
% OUTPUT : sqr - the desired Magic Square
% USAGE sqr = msqr_Bachet(n)
% EXAMPLE sqr = msqr_Bachet(3);

%Check that the input n for the order of the magic square is an odd number.
if mod(n,2)~= 1
    error('The input for the order of the Magic Square is not an Odd Number.');
```

end

```

sqr = zeros(n);
middle = 1+(n-1)/2;
% STEP 1: Star with the cell immediately above the middle cell and put 1 there.
% STEP 2: Move diagonally upwards from left to right placing numbers 2,3,4.. till it
% is not possible to do so anymore.
% STEP 3: When you reach the top row move to the bottom cell of the next column and apply step 2
% as far as possible.
% STEP 4: When you reach the extreme right column move to the row immediately above and go to
% the first cell in that row and fill it up with the next number,
% then apply step 2 as far as possible.
% STEP 5: When you reach a cell that is already occupied move 2 cells up
% in the same column and fill it with the next number

%STEP 1
currentRow = middle- 1;
currentColumn = middle;
sqr(currentRow,currentColumn) = 1;

k = 2;
while k <= n^2
    % APPLY STEP 2 as far as possible
    if currentRow-1>=1 & currentColumn+1<= n
        currentRow = currentRow-1 ;
        currentColumn = currentColumn+1 ;
        if sqr(currentRow,currentColumn) == 0
            sqr(currentRow,currentColumn) = k ;
            k = k+1;
        else
            currentRow = currentRow + 1 ;
            currentColumn = currentColumn-1 ;
            %APPLY STEP 5
            currentRow = msqr_moveUp2Rows(currentRow,n);
            sqr(currentRow,currentColumn) = k ;
            k = k+1;
        end
        continue;
    end
    %APPLY STEP 3
    if currentRow==1 & currentColumn<n
        currentRow = n;
        currentColumn = currentColumn+1 ;
        sqr(currentRow,currentColumn) = k ;
        k = k+1;

        continue;
    end
    %APPLY STEP 4
    if currentColumn == n & currentRow > 1
        currentRow = currentRow-1;
        currentColumn = 1 ;
        sqr(currentRow,currentColumn) = k ;
        k = k+1;
        continue;
    end
end

```

```

end
if currentColumn == n & currentRow == 1
    currentRow = n-1;
    sqr(currentRow,currentColumn) = k ;
    k = k+1;
    continue;
end
end

disp(num2str(sqr))

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function newrow = msqr_moveUp2Rows(row,n)
if row > 2 & row <= n
    newrow = row -2;
elseif row == 2
    newrow = n;
elseif row == 1
    newrow = n-1;
else
    error('Invalid inputs for moveUp2Rows.');
```

```

%FILE: msqr_diagonal
function centralSqr = msqr_diagonal(n)
% MSQR_DIAGONAL - MAGIC SQUARES OF ODD ORDER-III
%
% This function generates a magic square of odd order by the diagonal method
% (Chapter 4: Magic Squares of Odd Order-III)
%
% INPUT: n - the order of the magic square
% OUTPUT : sqr -the desired Magic Square
% USAGE sqr = msqr_diagonal(n)
% EXAMPLE sqr = msqr_diagonal(7);

if mod(n,2)~= 1
    error('The input for the order of the Magic Square is not an Odd Number. ');
end

m = (n - 1)/2;
bigSqrOrder = 2*n-1;
bigSqr = nan*zeros(bigSqrOrder);
middleRow = n;

currentColumn = 1;

for j=1:n
    currentRow = middleRow + j - 1;
    currentColumn = 1 + j - 1;
    for i=1:n
        bigSqr(currentRow,currentColumn)=(j-1)*n + i ;
        currentRow = currentRow-1;
        currentColumn = currentColumn + 1;
    end
end

mysqr = bigSqr;

centralSqrStartRow = m+1;
centralSqrEndRow = bigSqrOrder-m;
centralSqrStartCol = m+1;
centralSqrEndCol = bigSqrOrder-m;

centralSqr = bigSqr(centralSqrStartRow:centralSqrEndRow , centralSqrStartCol:centralSqrEndCol );

cutoutTop = bigSqr(1:centralSqrStartRow-1,centralSqrStartCol+1: centralSqrEndCol-1);
cutoutLeft = bigSqr(centralSqrStartRow+1:centralSqrEndRow-1,1 :centralSqrStartCol-1 );

cutoutRight = bigSqr(centralSqrStartRow+1:centralSqrEndRow-1 ,centralSqrEndCol+1:bigSqrOrder);
cutoutBottom = bigSqr( centralSqrEndRow+1:bigSqrOrder ,centralSqrStartCol+1: centralSqrEndCol-1 );

%paste cutoutRight
for row =1:size(cutoutRight,1)
    for col = 1:size(cutoutRight,2)
        if ~isnan(cutoutRight(row,col))
            centralSqr(1+row,col)= cutoutRight(row,col);
        end
    end
end

%paste cutoutLeft
for row =1:size(cutoutLeft,1)
    for col = 1:size(cutoutLeft,2)
        if ~isnan(cutoutLeft(row,col))
            centralSqr(1+row, col+m+1)= cutoutLeft(row,col);
        end
    end
end

%paste cutoutTop
for row =1:size(cutoutTop,1)

```

```
for col = 1:size(cutoutTop,2)
    if ~isnan(cutoutTop(row,col))
        centralSqr(row + m+1,col+1 )= cutoutTop(row,col);
    end
end
end

%paste cutoutBottom
for row =1:size(cutoutBottom,1)
    for col = 1:size(cutoutBottom,2)
        if ~isnan(cutoutBottom(row,col))
            centralSqr(row ,col+1 )= cutoutBottom(row,col);
        end
    end
end
end

disp(num2str(centralSqr))
```

```

%FILE msqr_dblEven1

function finalSqr = msqr_dblEven1(n)
% MSQR_DBLEVEN1 - MAGIC SQUARES OF DOUBLY EVEN ORDER-I
%
% This function generates a magic square of doubly even order
% (Chapter 5: Magic Squares of Doubly Even Order - I)
%
% INPUT: n - the order of the magic square
% OUTPUT : sqr -the desired Magic Square
% USAGE sqr = msqr_dblEven1(n)
% EXAMPLE sqr = msqr_dblEven1(8);
% SEE ALSO: msqr_dblEven1b

if mod(n,4)~=0
    error('The input for the order of the Magic Square is not a doubly even number.');
```

end

%This will create a n X n matrix, with the first row containing the entries 1 2 3 ...n;
% the second row running from (n+1) (n+2)...2n; and so on
firstSqr = reshape(1:n^2,n,n);

```

constMat = ones(n,n)*(n^2+1);
% we create a 4 by 4 matrix . All entres are either a 1 or 0;
% 1 if the element lies on one of the 2 principal diagonals and 0 otherwise
tile = eye(4) + flipud(eye(4));
% We now replicate this tile by the use of the repmat function.
tileMatrix = repmat(tile,n/4,n/4);

y = ( constMat -firstSqr ).* tileMatrix ;
z = firstSqr.*(~tileMatrix);
finalSqr = y+z;
disp(num2str(finalSqr))

% msqr_dblEven1b
function sqr = msqr_dblEven1b(n)
% MSQR_DBLEVEN1B - MAGIC SQUARES OF DOUBLY EVEN ORDER-I
%
% This function generates a magic square of doubly even order using 1 line of code
% (Chapter 5: Magic Squares of Doubly Even Order - I)
%
% INPUT: n - the order of the magic square
% OUTPUT : sqr -the desired Magic Square
% USAGE sqr = msqr_dblEven1b(n)
% EXAMPLE sqr = msqr_dblEven1b(16)
% SEE ALSO: msqr_dblEven1

sqr = (ones(n,n)*(n^2+1)-reshape(1:n^2,n,n)).*(repmat(eye(4)+
flipud(eye(4)),n/4,n/4))+reshape(1:n^2,n,n).*(~(repmat(eye(4)+flipud(eye(4)),n/4,n/4)));

% msqr_dblEven2
```

```

%FILE msqr_dblEven2
function finalSqr = msqr_dblEven2(n)
% MSQR_DBLEVEN2 - This function generates a magic square of doubly even order as
% described in Chapter 6 (Magic Squares of Doubly Even Order - II)
% INPUT: n - order of the magic square. The number n must be double even
% OUTPUT: finalSqr - the magic square of order n.
% USAGE sqr = msqr_dblEven2(n)
% EXAMPLE: sqr = msqr_dblEven2(8) will return a magic square of order 8.
%
% Author Saurabh Singal
% Date: 11 May, 2004.

%check that the order n is indeed a doubly even number
if mod(n,4)~=0
    error('The input for the order of the Magic Square is not a doubly even number.');
```

end

```

% STEP 1: Take a n by n square and proceed from left to right in the first row,
%         filling the cells with the numbers 1,2,3,...,n ;
%         (n+1),(n+2,...,2n in the second row , and so on for all the rows.
% STEP 2: Take the 4 small squares of order n/4 at the four corners and remove them
%         also remove the numbers of the central square of order n/2 at the centre of the square
% STEP 3: Fill up the remaining (n^2)/2 cells by the following rule -
%         to fill any blank cell (call it x) find the cell symmetric to with respect to the centre,
%         look up the number in this cell and fill this number in cell x.
%
sequence = 1:n^2; %STEP 1
%firstSqr will have numbers from n^2
% finalSqr will start as a copy of the firstSqr, then we will remove the corner squares and central
% square, and using as a reference, the firstSqr, will up the blank cells as per step 3
firstSqr = reshape(sequence,n,n)';
finalSqr = firstSqr;

smallSqrOrder = n/4;
centralSqrOrder = n/2;
% Replace element of the central part of the square (steps 2 and 3)
for row = (smallSqrOrder+1): (smallSqrOrder+centralSqrOrder)
    for col= (smallSqrOrder+1): (smallSqrOrder+centralSqrOrder)
        finalSqr(row,col) = firstSqr(n-row+1,n-col+1 );
    end
end

%top left small square (steps 2 and 3)
for row = 1:smallSqrOrder
    for col= 1:smallSqrOrder
        finalSqr(row,col) = firstSqr(n-row+1, n-col+1);
    end
end

%top right square (steps 2 and 3)
for row = 1: smallSqrOrder
    for col= (n-smallSqrOrder+1): n
        finalSqr(row, col) = firstSqr(n - row+1, n-col +1);
    end
end

%bottom left small square (steps 2 and 3)
for row = (n-smallSqrOrder+1): n
    for col= 1:smallSqrOrder
        finalSqr(row, col) = firstSqr(n-row+1,n-col+1);
    end
end

%bottom Right small square (steps 2 and 3)
for row = (n-smallSqrOrder+1): n
    for col= (n-smallSqrOrder+1): n
        finalSqr(row, col) = firstSqr(n-row+1,n-col+1);
    end
end

```

```
end
end
%display the magic square.
disp(num2str(finalSqr))
```

```

% FILE msqr_Kothari
function sqr = msqr_Kothari(n)
% msqr_Kothari - This function generates a magic square of doubly even order as
% described in Chapter 6 (Magic Squares of Doubly Even Order - II)
% INPUT: n - order of the magic square. The number n must be double even
% OUTPUT: sqr - the magic square of order n.
% USAGE: sqr = msqr_Kothari(n)
% EXAMPLE: sqr = msqr_Kothari(8) will return a magic square of order 8.
%
% Author Saurabh Singal
% Date: 11 May, 2004.

if mod(n,4)~=0
    error('The input for the order of the Magic Square is not a doubly even number.');
```

```

end

central = zeros(n,n);
top = zeros(n/2-1,n);
bottom = zeros(n/2-1,n);

augmented = [top;central;bottom];

startRow = size(top,1)+1;
startCol=1;
k=1;

row = startRow;
col = startCol;
% There are n/4 clockwise and n/4 anti-clockwise rotations;
% In each rotation 2n numbers are filled, so all n^2 (=2n*n/4 + 2n*n/4) numbers are filled.
% Each rotation has 4 legs, and therefore n/2 numbers filled each time.

for numRotations = 1: n/4
    % GO CLOCKWISE
    % Find the position to start clockwise
    col=1;
    row = min(find(augmented(n/2:end,col)==0)) + (n/2-1); % first row of the central square that is empty in the first column;
    for counter = 1 : n/2
        augmented(row,col) = k; row = row-1; col = col+1; k = k+1;
    end
    row=row+1;
    for counter = 1 : n/2
        augmented(row,col) = k; row=row+1; col=col+1; k = k+1;
    end
    col=col-1;
    for counter = 1 : n/2
        augmented(row,col) = k; row=row+1; col=col-1; k = k+1;
    end
    row=row - 1;
    for counter = 1 : n/2
        augmented(row,col) = k; row=row-1; col=col-1; k = k+1;
    end

    % GO ANTI-CLOCKWISE
    % Find the position to start Anti-clockwise
    col =n;
    row = min( find(augmented(n/2:end,n)==0)) + (n/2-1); % first row of the central square that is empty in the last column;
    for counter = 1 : n/2
        augmented(row,col)=k; row=row-1; col=col-1; k=k+1;
    end
    row = row + 1;
    for counter = 1 : n/2
        augmented(row,col)=k; row=row+1; col=col-1; k=k+1;
    end
    col= col+1;
    for counter = 1 : n/2
        augmented(row,col)=k; row=row+1; col=col+1; k=k+1;
    end
end

```

```
row = row -1;
for counter =1 : n/2
    augmented(row,col)=k; row=row-1; col=col+1; k=k+1;
end
end
bottom = augmented(3*n/2:end,:);
top = augmented(1:n/2-1,:);
central = augmented(n/2:3*n/2-1,:);
%Paste together the n/2-1 rows of the "bottom" and the same from the "top",
%separated by 2 rows of zeros so that we have a n X n square matrix
% We will add this to the "central" square
BottomTop = [ bottom ;zeros(2,n); top ];
sqr = BottomTop +central;
disp(num2str(sqr))
```

```

%FILE msqr_Kaumudi
function finalSqr = msqr_Kaumudi(n)
% MSQR_KAUMUDI: Generate a Magic Square of a Singly Even Order
% Chapter 8, Magic Square of Singly Even Order-I
% This uses the method of Pandit Narayana's Ganita Kaumudi
%
% INPUT: n - the order of the magic square
% OUTPUT : sqr -the desired Magic Square
% USAGE sqr = msqr_Kaumudi(order)
% EXAMPLE sqr = msqr_Kaumudi(6);

if ~(mod(n,4)~=0 & mod(n,2)==0)
    error('The input for the order of the Magic Square is not a Singly Even number.');
```

end

```

m = (n/2-1)*(1/2);
middle = n/2;
slisthaMatrix = zeros(n,n);
finalSqr = reshape(1:n^2,n,n)';

for i =2:2:n
    finalSqr(i,:)=fliplr(finalSqr(i,:));
end

%Mark off the Slistha cells
for rownum =1:n
    k=1;
    slisthacount=0;
    while slisthacount<m
        if rownum ~=k & rownum ~= n-k+1
            slisthaMatrix(rownum,k)=1;
            slisthaMatrix(rownum,n-k+1)=1;
            slisthacount = slisthacount+1;
        end
        k=k+1;
    end
end
% disp(num2str(slisthaMatrix))

%step (i): exchange the right most middle two slistha square entries
% with the entries in the same rows and middle columns
a = finalSqr(middle,n) ;
b = finalSqr(middle+1,n) ;
bprime =finalSqr(middle+1, middle);
aprime =finalSqr(middle , middle);

finalSqr(middle,n) = aprime;
finalSqr(middle+1,n) = bprime;
finalSqr(middle ,middle)= a;
finalSqr(middle+1, middle ) = b;

%step (ii): Take the upper half slistha square entries and interchange with
% the symmetrically placed entries in the lower half

for rownum = 1 : n/2
    for colnum = 1:n
        if(slisthaMatrix(rownum,colnum))
            upperEntry = finalSqr(rownum,colnum);
            lowerEntry = finalSqr(n+1-rownum,colnum);
            finalSqr(rownum,colnum) =lowerEntry;
            finalSqr(n+1-rownum,colnum) = upperEntry;
        end
    end
end

%step(iii): Take the three rectangles and rotate
```

```
%them anti clockwise about the centre of the square
rectangle1 =finalSqr(1:2*m,2*m+2) ;
rectangle2 =finalSqr(1:2*m,2*m+1) ;
rectangle3 =finalSqr(2*m+3:n,2*m+2) ;

finalSqr(1:2*m,2*m+2) = flipud(rectangle3); %rectangle1 is replaced by rectangle3
finalSqr(1:2*m,2*m+1) =rectangle1 ; %rectangle2 is replaced by rectangle1
finalSqr(2*m+3:n,2*m+2) = flipud(rectangle2); %rectangle3 is replaced by rectangle2

fprintf('\n\nThe Magic Square of Order %d generated using Pandit Narayan"s method described in Ganita Kaumudi\n',n);
disp(num2str(finalSqr))
```

```

%FILE msqr_Strachey(n)
function finalSqr = msqr_Strachey(n)
% MSQR_STRACHEY - MAGIC SQUARES OF SINGLY EVEN ORDER-II
%
% This function generates a magic square of singly even order
% (Chapter 9: Magic Squares of Singly Even Order - II)
%
% INPUT: n - the order of the magic square
% OUTPUT : sqr - the desired Magic Square
% USAGE sqr = msqr_Strachey(n)
% EXAMPLE sqr = msqr_Strachey(6);
if ~(mod(n,4)~=0 & mod(n,2)==0)
    error('The input for the order of the Magic Square is not a Singly Even number.');
```

end

```

m = (n/2-1)*(1/2);
U = n/2;
middle =m+1;

%STEP 1
fprintf('\n\nGenerating the squares of order %d by the Narayana method (shown below)\n\n',n,n/2);
sqrA = msqr_Narayana(n/2);
% no need to generate squares B, C, D from scratch;
% merely add U^2 to each element of sqrA to get square B etc
sqrB = sqrA + (U)^2;
sqrC = sqrB + (U)^2;
sqrD = sqrC + (U)^2;

%STEP 2: In the middle row of square A, leave the first cell as it is
% and interchange the numbers in the next m cells with corresponding numbers of square D
for j=2:2+m-1
    amiddlej = sqrA(middle,j);
    dmiddlej = sqrD(middle,j);
    sqrA(middle,j) = dmiddlej;
    sqrD(middle,j) = amiddlej;
end

%STEP 3: In each row of square A, except the middle row, take the first m cells
% from the left and interchange with corresponding entries of square D
for i = 1 : n/2
    if i ~= middle
        for k=1:m
            aik = sqrA(i,k);
            dik = sqrD(i,k);
            sqrA(i,k) = dik;
            sqrD(i,k) = aik;
        end
    end
end

% FINAL STEP( STEP 4):
% Take the m-1 columns of square C from the extreme right and and
% exchange all the entries in these columns with corresponding entries of square B
for v = U-m+2 : U
    for w = 1:U
        cwv = sqrC(w,v);
        bwv = sqrB(w,v);
        sqrC(w,v) = bwv;
        sqrB(w,v) = cwv;
    end
end

%combine the 4 quadrants to get the answer
finalSqr= [ sqrA sqrC ; sqrD sqrB];

fprintf('\n\nMagic Square of Order %d Generated by the Method of Ralph Strachey\n',n);
disp(num2str(finalSqr))

```

```

%FILE msqr_ConwayLUX
function newsqr = msqr_ConwayLUX(n)
% MSQR_CONWAYLUX: Generate a Magic Square of a Singly Even Order
% Chapter 10, Magic Square of Singly Even Order-III
% This uses the method of Pandit Narayana's Ganita Kaumudi
%
% INPUT: n - the order of the magic square
% OUTPUT : sqr - the desired Magic Square
% USAGE sqr = msqr_ConwayLUX(n)
% EXAMPLE sqr = msqr_ConwayLUX(10);

if ~(mod(n,4)~=0 & mod(n,2)==0)
    error('The input for the order of the Magic Square is not a Singly Even number.');
```

end

```

m = (n/2-1)*(1/2);

fprintf('msqr_ConwayLUX: Displaying the Centres of the Square of order %d X %d generated by Narayan Method\n',n/2,n/2)
ndllsq = msqr_Narayana(n/2);

% The methodsq is a matrix of L's, U's, and X's which will tell us which
% tranformation (L, U or X) to apply
methodsq = repmat('O',n/2,n/2);%create a matrix of O's
methodsq(1:m+1,:)= 'L'; % m+1 rows of L : STEP 2(i)
methodsq(m+2 ::)= 'U'; % a single row of U's: STEP 2(ii)
methodsq(m+3 :n/2,:)= 'X'; %rest are rows of X: STEP 2(iii)

%STEP 2(iv): Interchange the middle U with L above it
methodsq(m+1,m+1)= 'U';
methodsq(m+2,m+1)= 'L';

fprintf('This is the square of transformations :\n');
disp(methodsq)
newsqr=zeros(n,n);
dummy=zeros(2,2);

for i = 1: n/2
    for j =1: n/2
        %STEP 3: using the centres of the square filled by Narayan method,
        %get the small 2 by 2 sqr
        dummy(1,1) = 4*ndllsq(i,j)-3;
        dummy(1,2) = 4*ndllsq(i,j)-2;
        dummy(2,1) = 4*ndllsq(i,j)-1;
        dummy(2,2) = 4*ndllsq(i,j) ;

        switch methodsq(i,j) %STEP 4: apply the transofmation L, U or X as may be appropriate
            case 'L'
                newsqr(2*i-1:2*i,2*j-1:2*j)= Ltransformation (dummy);
            case 'U'
                newsqr(2*i-1:2*i,2*j-1:2*j)= Utransformation (dummy);
            case 'X'
                newsqr(2*i-1:2*i,2*j-1:2*j)= Xtransformation (dummy);
        end
    end
end
fprintf('\n\nThe Magic Square of Order %d generated using Conway"s LUX method\n',n)
disp(num2str(newsqr))
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function L =Ltransformation(orgsqr)
% Apply the L transformation
L=zeros(2,2);
L(1,1)= orgsqr(2,2);
L(1,2)= orgsqr(1,1);
L(2,1)= orgsqr(1,2);
```



```

%FILE msqr_verify

function ch = msqr_verify(sqr)
% VERIFY that the given matrix is a magic square by
% checking that the sum of the column, sum of the rows and the sum of diagonals are
% identical
%USAGE ch = msqr_verify(sqr);

if size(sqr,1)~= size(sqr,2)
    error('Input is not a square matrix');
end
columnSums= sum(sqr);
rowSums = sum(sqr,2);
sumDiag1 = sum(diag(sqr));
sumDiag2 = sum(diag(fliplr(sqr))) ;
fprintf('The sum of the rows is \n')
for i = 1:length(rowSums)
    fprintf('Row No. %d, Sum = %d\n',i, rowSums(i) );
end

fprintf('The sum of the columns is \n',columnSums );
for j = 1:length(columnSums)
    fprintf('Column No. %d, Sum = %d\n',j, columnSums(j) );
end
fprintf('The sum of the Principal Diagonal (Top Left to Bottom Right) is %d\n',sumDiag1);
fprintf('The sum of the Principal Diagonal (Top Right to Bottom Left) is %d\n', sumDiag2);

checkRowSum =all(sum(sqr) ==sum(diag(sqr)));
checkColumnSum = all(sum(sqr,2) ==sum(diag(sqr)));
if checkRowSum == 1 & checkColumnSum ==1 & sumDiag1==sumDiag2
    ch = 1;
else
    ch=0;
end

if ch==1
    fprintf('\nVerified that the input is a magic square.\n');
else
    fprintf('\nThe input is NOT a magic square.\n');
end

```

```

%msqr_demo
clc
fprintf('*****\n');
fprintf('*****\n');
fprintf('Magic Squares of Odd Order-I: The Narayana-De la Loubre Method, Chapter 2.\n');
fprintf('*****\n');
fprintf('*****\n');
fprintf('Generating Magic Square of Order 3 using the Narayana Method\n');
sqr = msqr_Narayana(3);
inpstr = input('Press ENTER to continue...\n');
fprintf('Verifying that we generated a Magic Square\n');
ch=msqr_verify(sqr);
inpstr = input('Press ENTER to continue...\n');
fprintf('===== \n');
fprintf('===== \n');

fprintf('Generating Magic Square of Order 5 using the Narayana Method\n');
sqr = msqr_Narayana(5);
inpstr = input('Press ENTER to continue...\n');
fprintf('Verifying that we generated a Magic Square\n');
ch=msqr_verify(sqr);
inpstr = input('Press ENTER to continue...\n');
fprintf('===== \n');
fprintf('===== \n');

fprintf('Generating Magic Square of Order 7 using the Narayana Method\n');
sqr = msqr_Narayana(7);
inpstr = input('Press ENTER to continue...\n');
fprintf('Verifying that we generated a Magic Square\n');
ch=msqr_verify(sqr);
inpstr = input('Press ENTER to continue...\n');
fprintf('===== \n');
fprintf('===== \n');

fprintf('*****\n');
fprintf('*****\n');
fprintf('Magic Squares of Odd Order-II: Bachet's Method, Chapter 3.\n');
fprintf('*****\n');
fprintf('*****\n');

fprintf('Generating Magic Square of Order 7 using Bachet's Method\n');
sqr = msqr_Bachet(7);
inpstr = input('Press ENTER to continue...\n');
fprintf('Verifying that we generated a Magic Square\n');
ch=msqr_verify(sqr);
inpstr = input('Press ENTER to continue...\n');
fprintf('===== \n');
fprintf('===== \n');
fprintf('Generating Magic Square of Order 9 using Bachet's Method, Chapter 3\n');
sqr = msqr_Bachet(9);
inpstr = input('Press ENTER to continue...\n');
fprintf('Verifying that we generated a Magic Square\n');
inpstr = input('Press ENTER to continue...\n');
ch=msqr_verify(sqr);
fprintf('===== \n');
fprintf('===== \n');

fprintf('*****\n');
fprintf('*****\n');
fprintf('Magic Squares of Odd Order-III: Diagonal Method, Chapter 4.\n');
fprintf('*****\n');
fprintf('*****\n');
fprintf('Generating Magic Square of Order 7 using the Diagonal Method\n');
sqr =msqr_diagonal(7);
inpstr = input('Press ENTER to continue...\n');
fprintf('Verifying that we generated a Magic Square\n');
ch=msqr_verify(sqr);
inpstr = input('Press ENTER to continue...\n');

```

```

fprintf('=====\n');
fprintf('=====\n');
fprintf('Generating Magic Square of Order 9 using the Diagonal Method\n');
sqr =msqr_diagonal(9);
fprintf('Verifying that we generated a Magic Square\n');
ch=msqr_verify(sqr);
fprintf('=====\n');
fprintf('=====\n');
fprintf('*****\n');
fprintf('*****\n');
fprintf('Magic Squares of Doubly Even Order-I, Chapter 5.\n');
fprintf('*****\n');
fprintf('*****\n');
fprintf('Generating Magic Square of Order 4 using msqr_dblEven1 (Method of Chapter 5)\n');
sqr = msqr_dblEven1(4) ;
fprintf('Verifying that we generated a Magic Square\n');
inpstr = input('Press ENTER to continue....\n');
ch=msqr_verify(sqr);
fprintf('=====\n');
fprintf('=====\n');

fprintf('Generating Magic Square of Order 8 using msqr_dblEven1 (Method of Chapter 5)\n');
sqr = msqr_dblEven1(8) ;
fprintf('Verifying that we generated a Magic Square\n');
inpstr = input('Press ENTER to continue....\n');
ch=msqr_verify(sqr);
fprintf('=====\n');
fprintf('=====\n');
fprintf('=====\n');
fprintf('=====\n');
fprintf('*****\n');
fprintf('*****\n');
fprintf('Magic Squares of Doubly Even Order-II, Chapter 6.\n');
fprintf('*****\n');
fprintf('*****\n');
fprintf('Generating Magic Square of Order 4 using msqr_dblEven2 (Method of Chapter 6)\n');

sqr = msqr_dblEven2(4) ;
fprintf('Verifying that we generated a Magic Square\n');
inpstr = input('Press ENTER to continue....\n');
ch=msqr_verify(sqr);
fprintf('=====\n');
fprintf('=====\n');

fprintf('Generating Magic Square of Order 8 using msqr_dblEven2 (Method of Chapter 6)\n');
sqr = msqr_dblEven2(8) ;
fprintf('Verifying that we generated a Magic Square\n');
ch=msqr_verify(sqr);
inpstr = input('Press ENTER to continue....\n');
fprintf('=====\n');
fprintf('=====\n');
fprintf('*****\n');
fprintf('*****\n');
fprintf('Magic Squares of Doubly Even Order-III, Chapter 7.\n');
fprintf('*****\n');
fprintf('*****\n');
fprintf('*****\n');
fprintf('*****\n');
fprintf('Generating Magic Square of Order 12 using Prof L.S. Kothari"s method (Chapter 7)\n');
sqr = msqr_Kothari(12) ;
fprintf('Verifying that we generated a Magic Square\n');
ch=msqr_verify(sqr);
inpstr = input('Press ENTER to continue....\n');

fprintf('Magic Squares of Singly Even Order-I, Chapter 8.\n')
fprintf('*****\n');
fprintf('*****\n');

```

```
fprintf('Verifying that we generated a Magic Square\n');
ch=msqr_verify(sqr);
inpstr = input('Press ENTER to continue....\n');
fprintf('=====\n');
fprintf('=====\n');
fprintf('*****\n');
fprintf('*****\n');
fprintf('Magic Squares of Singly Even Order-II, Chapter 9.\n');
fprintf('*****\n');
fprintf('*****\n');
fprintf('Generating Magic Square of order 6 by Strachey"s Method\n') ;
sqr = msqr_Strachey(6);
fprintf('Verifying that we generated a Magic Square\n');
ch=msqr_verify(sqr);
inpstr = input('Press ENTER to continue....\n');
fprintf('=====\n');
fprintf('=====\n');
fprintf('*****\n');
fprintf('*****\n');
fprintf('Magic Squares of Singly Even Order-III, Chapter 10.\n');
fprintf('*****\n');
fprintf('*****\n');
fprintf('Generating Magic Square of order 10 by Conway"s LUX Method\n') ;
sqr = msqr_ConwayLUX(10);
fprintf('Verifying that we generated a Magic Square\n');
ch=msqr_verify(sqr);
inpstr = input('Press ENTER to continue....\n');
fprintf('=====\n');
fprintf('=====\n');
```