Question 4. Whack-a-Mole

Task: For a given n by n board configuration write algorithm which gives the sequence of holes to be hit in order to empty the board of all moles. Effectively solving a linear algebra problem, with layout defined below using the example of a 4 by 4 board.

Setup/ problem defintion:
s- starting position vector (16x1) of moles this should contain 1 when mole present and zero when mole absent in the location corresponding to index *i* on the board.

x- vector (16x1) defining the holes which need to be whacked to arrive at empty state, which terminates the game. The i'th element of x is equal to 1 if hole is to be hit and 0 otherwise.
note: sequence in which hits are performed does not matter as long as all the hits are executed.

A-is a 16 by 16 matrix (n^2 by n^2), which contains a representation of all the possible outcomes of whacking a hole. Each column of **A** corresponds to a hole (i) and each row corresponds to the hole that is affected when i is whacked. So for instance A(all rows in column 1) corresponds to the response of whacking hole at position 1 and is [1 1 0 0; 1 0 0 0; 0 0 0 0; 0 0 0 0].

Notation: *i* will generally be used to refer to rows and *j* to refer to column counters.

It should be noted that a 4 by 4 matrix of holes can be represented as a 16 by 1 vector, indexed like on the right.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

For the example, for a board layout given in homework, the following vectors could then be defined: *(indices assigned from top left to bottom right)*

s=(0,1,0,0,1,1,1,0,0,0,0,1,1,1,0,1)
x=(0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,1)
note. X is the least amount of moves solution however other valid solutions also exist. i.e. *x*=(0,0,1,1,0,0,0,0,1,1,0,1,0,0,0,0), where mod(A*x,2)=s as well. This is because the A matrix rank for this example is 14 meaning there are 2 solutions as the system of 16 equation is not fully linearly independent)

A is always composed out of sub-matrices, building blocks of 3 types of fundamental matrices (I,O and Z as per illustration below). Depending on the size of the problem these -sub matrices will have dimensions of n by n.

For a 4 by 4 problem

O is the zero matrix

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

I is the identity matrix

1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

Z is:

1	1	0	0
1	1	1	0
0	1	1	1
0	0	1	1

Matrix **A** (16 x16) can be represented as a 4x4 matrix of:

Z	I	O	O
I	Z	I	O
O	I	Z	I
O	O	I	Z

The construction of A always follows this pattern regardless of the size of the n.
So for any A

Z(1,1)	I(1,2)	O(1,3)	..	O(1,n)
I(2,1)	Z(2,2)	I(2,3)	..	O(2,n)
O(3,1)	I(3,2)	Z(3,3)	..	.
..	….	…	…..	I(n-1,n)
O(n,1)	O(n,2)	….	I(n,n-1)	Z(n,n)

The first Column contains the board configuration after hitting hole a(1,1) in a vectorised form and assuming no other moles present at time of hitting. The second column corresponds to the pattern of 1's and 0's after hitting hole a(1,2) and so on.

It is also interesting to observe that if a hole is whacked an even amount of times it is equivalent to not whacking it at all, and if on the other hand, it is whacked for an odd amount of times that counts. Thus solving the whack-a-mole problem boils down to solving a system in Modulo 2 where only the odd reminder counts. In order to find vector x it is need to solve: $\mathbf{A}x=s$ (where s and x are vectors and **A** is a matrix) .This should be done in in a GF2 system, which stands for the Galois Field with modulo 2.

A solution for $\mathbf{A}x=s$ can be found by performing gaussian elimination on matrix **A** augmented with the vector s and using the property that 1+1=0 (XOR). Broadly, gaussian elimination consists of 3 allowable operations: 1)replace row by a constant multiple of the row, 2) replace a row by adding a constant multiple of another row to it or 3) swap two rows. The iteration of these operations ends when the system is reduced to the row echelon form of matrix A (matrix of zeros only except for 1's on the diagonal, with first 1 in the top left corner).

Consideration should also be given to whether a solution to the system $\mathbf{A}x=s$ exists. For this the rank of a matrix needs to be defined; this value represents the number of linearly independent rows or columns of a matrix. The rank of a matrix A is the number of non-zero rows in the row reduced form of matrix A.

In a 3 by 3 system the rank of **A** is always maximal (equivalent to 9=n*n) and equal to the rank of the augmented matrix [As] so there exists only one unique solution. And hence system is always solvable. For larger boards the rank of A is not maximal, meaning that there is linear dependence in the rows/columns of A and that more than one solution or no solutions exist. For instance, in the 4 by 4 case rank is 14 and in the 5 by 5 case rank is 23. For cases where n>3 it is easy to check if solution exists after row reduction has been performed and matrix A is in row echelon form. The condition which needs to be satisfied is that the rank of the resultant augmented matrix [As] needs to be larger than or equal to the rank of resulting echelon form matrix A.
I.e. after row reduction : rank(As)>=rank(A). This condition has been tested on the example given in the homework assignment. In this case rank(A) is 14 and rank(As) is 15.

Note:

An alternative way to check if system has a solution would be to calculate the null space vectors of A and take the dot product of these with vector s. If s is orthogonal to the null space vector (i.e.dot product is equal to zero) then a unique solution exits. If however the dot product is non zero, this means that the board configuration has no solution.

PSEUDOCODE:

Initialise problem:
For a given board configuration construct vector *s* and matrix **A** as per description on page 1, this is prior to implementing algorithm below. I assume I do not need to define how the can be achieved computationally (i.e. isn pseudocode).

1st Step. Determine if board can be cleared

```
if n<3
        display('game has no sense?')
        return
elseif  n= 3
        board can be cleared move to step 2

else    for systems with n>3 need to proceed to step 2 first and only check if solution exists after
        performing gaussian elimination.
```

2nd Step. Find path to clear board of all moles (if solution exists)

Equivalent to performing gaussian elimination down to the row echelon form of an augmented matrix A.

Start by joining vector s to matrix A as an additional last column numbered j=(n^2)+1. Thus effectively creating a new matrix [AIs] of size (n^2) by (n^2)+1, called the augmented matrix.
In pseudocode:

```
As = [A s];
```

The first part of this code orders the rows so that the rows containing the left-most leading ones move to the top:

```
for j=1:(n^2) (where j corresponds to columns of matrix A who's augmented size is n+1, but we
        do not wish to loop over age last column here)
    k=j
    while (As(k,j)==0     (equivalent to searching each column for the first instance of 1)
        k=k+1
    end while
    when found swap row As(j,:) with As(k,:) correct
```

Next part of code makes sure there is only one 1 on the diagonal of the matrix and if this is not the case it performs XOR row operations until such a form is obtained. When form is obtained algorithm jumps to next column by updating the j counter.

```
    for i=1:(n^2) (where i corresponds to rows)
        if  ((i not equal j) and (As(i,j)=1))
        for ii=j:(n^2)+1 (this counter should extend to n+1 as XOR operations also need to
                be performed on the last column, which upon completion of iterations
                will give the final solution i.e.. vector x)
        As(i,ii)=As(i,ii) XOR As(j,ii)   (does XOR logical addition/subtraction, where 1+1=0)
        end for
        end if
    end for
    end for
```

3rd Step. Check if the row reduced augmented matrix [As] has a solution

```
if (rank(As) < rank(A)
        display(‘ system has no solution')
        return
end if
```

I assume I do not need to write the algorithm to find the rank of a matrix. However for completeness, the logic would be as flows:

```
for i=1:n^2
      if A(i,:)==1 (note A is already in row reduced form)
            rank=rank+1   (for each non-zero row increase rank by 1)
      end if
end for
```

COMPLEXITY

An algorithm is solvable in polynomial time if the number of steps required to complete the algorithm for a given input is O(n^k) for some nonnegative integer k, where n is the complexity of the input. In the whack-a-mole example the complexity n is the sure of the size of the board so n=n^2, here the power of two only stems from the definition of n. Further to that the computational expense of the algorithm presented above can be determined by looking at the sections of the algorithm; the pivoting/ or row shaping section and the XOR operation section form am loop which is O(n^3) complexity as we are iterating over n 3 in a nested loop. Therefore the total polynomial complexity in expressed as a function of the board size n, is O(n^6).