#include<iostream>

using namespace std;

int grid,dbg;

class node{

    public:

        //static int n;

        int n;

        int \*\*board;

        int curr\_row;

        node \*next;

        node(){

            //intial

            n=grid;

            next=NULL;

            curr\_row=-1;

            board = new int\*[n];

            for(int i=0;i<n;i++){

                board[i] = new int[n];

                for(int j=0;j<n;j++)

                    board[i][j] = 0;

            }

        }

        node(node \*parent){

            this->n=grid;

            this->curr\_row=parent->curr\_row+1;

            this->board=new int\*[n];

            for(int i=0;i<n;i++){

                this->board[i] = new int[n];

                for(int j=0;j<n;j++)

                    this->board[i][j] = parent->board[i][j];

            }

            this->next=NULL;

        }

        ~node(){

            for(int i=0;i<n;i++)

                delete board[i];

            delete board;

        }

        int place(int col){

            if(curr\_row == grid)

                return 0;

            int f=1;

            //left diag

            for(int r=curr\_row-1,c=col-1;r>=0 && c>=0;r--,c--){

                if(board[r][c])

                    f=0;

            }

            //right diag

            for(int r=curr\_row-1,c=col+1;r>=0 && c<n;r--,c++){

                if(board[r][c])

                    f=0;

            }

            //up

            for(int r=curr\_row-1;r>=0;r--){

                if(board[r][col])

                    f=0;

            }

            board[curr\_row][col]=1;

            return f;

        }

        void attacked(int c){

            int ct=0;

            char c1 = 'a'+c;

            char c2;

            c2 = 'a'+c;

            for(int i=1;curr\_row-i>=0;i++){//north

                if(board[curr\_row-i][c]){

                    cout << "Q" << c1 << n-curr\_row << " attacked by Q" << c2 << (n-curr\_row)+i << endl;

                }

            }

            c2 = 'a'+c;

            for(int i=1;curr\_row-i >= 0 && c+i < n;i++){//east-north

                c2++;

                if(board[curr\_row-i][c+i]){

                    ct++;

                    cout << "Q" << c1 << n-curr\_row << " attacked by Q" << c2 << (n-curr\_row)+i << endl;

                }

            }

            c2 = 'a'+c;

            for(int i=1;curr\_row-i >= 0 && c-i >= 0;i++){//west-north

                c2--;

                if(board[curr\_row-i][c-i]){

                    ct++;

                    cout << "Q" << c1 << n-curr\_row << " attacked by Q" << c2 << (n-curr\_row)+i << endl;

                }

            }

        }

        void print(){

            char file= 'a';

            for(int i=0;i<=n;i++){

                if(i==n){

                    cout << "   ";

                    for(int j=0;j<n;j++){

                        cout << file++ << " ";

                    }

                    cout << endl;

                }else{

                    for(int j=0;j<n;j++){

                        if(j == 0){

                            cout << n-i << " ";

                            if((n-i)%10 == (n-i))

                                cout << " ";

                        }

                        if(board[i][j])

                            cout << 'Q' << " ";

                        else if( (i+j)%2 )

                            cout << (char)176 <<" ";

                        else

                            cout << (char)178 <<" ";

                    }

                    cout << endl;

                }

            }

        }

};

class stack{

    public:

        int size;

        node \*head;

        stack(){

            size=0;

            head=NULL;

        }

        void push(node \*pnn){

            size++;

            if(head==NULL)

                head=pnn;

            else

                pnn->next=head;

                head=pnn;

        }

        node \*pop(){

            size--;

            node \*pnn = head;

            head = pnn->next;

            return pnn;

        }

};

void generate\_child(node \*curr,stack \*S,stack \*sol){

    node \*\*child = new node\*[grid];

    for(int i=0;i<grid;i++){

        child[i] = new node(curr);

        if(child[i]->place(i)){

            if(child[i]->curr\_row == grid-1)

                sol->push(child[i]);

            else

                S->push(child[i]);

        }else{

            if(dbg){

                cout << "<<deleted node>>" << endl;

                child[i]->print();

                child[i]->attacked(i);

                cout << "<<<<<<<<>>>>>>>>" << endl;

            }

            delete child[i];

        }

    }

}

void solve(stack \*S,stack \*sol){

    node \*curr,\*pnn;

    //intial

    pnn = new node;

    S->push(pnn);

    while(S->head!=NULL){

        curr=S->pop();

        generate\_child(curr,S,sol);

        delete curr;

    }

}

int main(){

    stack \*S=new stack,\*solution=new stack;

    cout << "enter grid size: ";

    cin >> grid;

    cout << "show deleted nodes 1/0 ?: ";

    cin >> dbg;

    // node::n = grid;

    solve(S,solution);

    cout << "<<<<<<< SOLUTION >>>>>>>" << endl;

    node \*trav = solution->head;

    while(trav != NULL){

        trav->print();

        cout << endl;

        trav = trav->next;

    }

    cout << solution->size << " solutions !!" << endl;

    cout << "<<<<<<<<<<<<>>>>>>>>>>>>" << endl;

    delete S;

    delete solution;

    return 0;

}