問２　Ｎクイーン問題に関する次の記述を読んで，設問１～３に答えよ。

Ｎクイーン問題とは，N×N マスの盤上で互いの利き筋に当たらないようなＮ個のクイーンの配置を見つける問題である。クイーンは，縦・横・斜めのいずれか一方向にどこまでも移動することができ，一度に移動できる範囲をクイーンの利き筋という。8×8 マスの盤上の行５列６に配置したクイーンの利き筋を，図１に示す。また，8×8 マスの場合のＮクイーン問題の解の一つを図２に示す。

なお，Ｎクイーン問題の解は存在しないこともあるし，複数存在することもある。

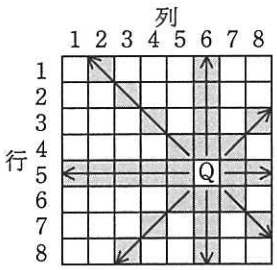

図１　クイーンの利き筋

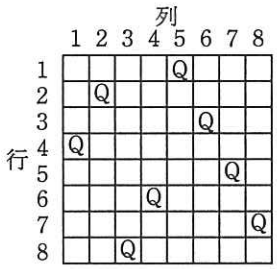

図２　8×8マスの解の例

注記　マス上の“Q”は
クイーンの位置を表す。

Ｎクイーン問題に対し，空の盤上にクイーンを配置し，その配置したクイーンの利き筋に当たらない位置を探索しながら，行順にクイーンを配置するという，次のような解法を考えた。

〔Ｎクイーン問題の解法〕

・１行目において，１列目にクイーンを配置する。次にこの１行目のクイーンの利き筋に当たらない２行目の列を１列目から順に探索し，クイーンを配置する。同様に次の行以降も，既に配置したクイーンの利き筋に当たらない列を探索し，クイーンを配置する。

・Ｎ行目までクイーンが配置できた場合は，解の一つが見つかったとして終了する。

・ある行でクイーンが配置できる列が見つからなかった場合は，一つ前の行に戻り，その行のクイーンを取り除く。取り除いたクイーンの次の列以降で，クイーンが配置できる列を探索する。それでも列が見つからなかった場合は，更に前の行に戻り，同様に繰り返す。

・１行目においてもクイーンが配置できる列がなくなった場合は，このＮクイーン問題の解はないということで終了する。

〔利き筋の判定〕

行 i 列 k のマスが既に配置したクイーンの利き筋に当たるか否かを容易に判別できるよう，盤面の利き筋の方向別に配列 col（列方向），upwd（斜め上方向）及び downwd（斜め下方向）を用意した（図 3～5）。解法では，一つの行には一つしかクイーンが配置されないので，行方向の判別は行う必要がない。

各配列の要素の値は，その方向にまだクイーンが配置されていないとき FREE となり，既に配置されているとき NOT_FREE となる。各要素の初期値は FREE である。

図 3～5 の矢印の先の番号は，各配列の添字に対応する。N×N マスの場合，配列 col の大きさは N であり，upwd と downwd の大きさはともに [ア] である。

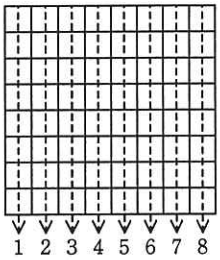

図3　列方向の配列
col（8×8マスの場合）

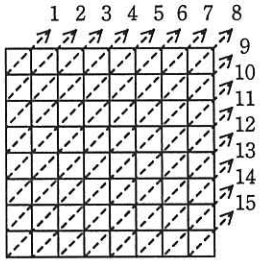

図4　斜め上方向の配列
upwd（8×8マスの場合）

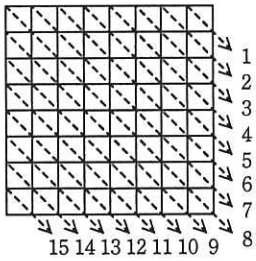

図5　斜め下方向の配列
downwd（8×8マスの場合）

例えば，図 6 のように 8×8 マスの盤上の行 1 列 1 と行 2 列 3 のマスにクイーンを配置した場合は，col[1]，col[3]，upwd[1]，upwd[4]，downwd[7] 及び downwd[8]の値が NOT_FREE となる。一般に N×N マスの盤上の行 i 列 k のマスにクイーンを配置した場合は，col[k]，upwd[i＋k－1] 及び downwd[[イ]]の値が NOT_FREE となる。

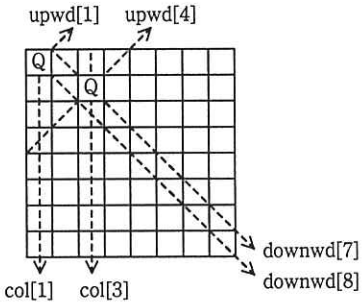

図6　クイーンの配置と利き筋の例（8×8 マスの場合）

〔N クイーン問題の解法のプログラム〕

i 行目以降についてクイーンの配置の仕方を探索する再帰関数 search のプログラムを図 7 に，メインプログラムを図 8 に示す。

関数 search は i 行目以降のクイーンの配置の仕方が見つかった場合に SUCCESS を，見つからなかった場合に FAILURE を戻す。

配列 pos は，行番号を添字とし，その行に配置したクイーンの位置（列番号）を値とする。配置されていない場合の値は 0 である。

```
function search(i)
    for ( [ ウ ] を [ エ ] から [ オ ] まで 1 ずつ増やす )
        if ( col[k]と upwd[i+k−1]と downwd[ [ イ ] ]が全て FREE と等しい) then
            // クイーンを配置する
            [ カ ]
            col[k] ← NOT_FREE
            upwd[i+k−1] ← NOT_FREE
            downwd[ [ イ ] ] ← NOT_FREE
            if ( i と N が等しい ) then
                return SUCCESS
            else
                if ( search( [ キ ] ) と SUCCESS が等しい ) then
                    return SUCCESS
                else
                    // クイーンを取り除く
                    pos[i] ← 0
                    col[k] ← FREE
                    upwd[i+k−1] ← FREE
                    downwd[ [ イ ] ] ← FREE
                endif
            endif
        endif
    endfor
    // クイーンが配置できる列が見つからなかった
    return FAILURE
endfunction
```

① （pos[i] ← 0 から downwd[[イ]] ← FREE）

図 7　関数 search のプログラム

```
// メインプログラム
配列 pos を初期化する
配列 col, upwd 及び downwd を初期化する
if ( search( [ ク ] ) と SUCCESS が等しい ) then
    解となるクイーンの配置を印字する
else
    "解が見つからなかった" と印字する
endif
```

図 8　メインプログラム

設問 1　N×N マスの場合，本文中の［　ア　］に入れる適切な字句を答えよ。

設問 2　N×N マスの場合，本文及び図 7 中の［　イ　］に入れる適切な字句を答えよ。

設問 3　〔N クイーン問題の解法のプログラム〕について，(1)～(3)に答えよ。

(1)　図 7 中の［　ウ　］～［　キ　］に入れる適切な字句を答えよ。

(2)　図 8 中の［　ク　］に入れる適切な字句を答えよ。

(3)　4×4 マスの場合，このプログラムによる解を図 9 に示す。この結果が得られるまでに，図 7 中の①の部分は何回実行されるか答えよ。

行＼列	1	2	3	4
1		Q		
2				Q
3	Q			
4			Q	

図 9　4×4 マスの解