

問 3 魔方陣に関する次の記述を読んで、設問 1~3 に答えよ。

魔方陣とは、正方形のマス目（方陣）に数を配置し、縦・横・対角線のいずれにおいても、その並びの数の合計が同じになるものである。ここでは、 $N \times N$ の方陣（ N は 3 以上の自然数）に 1 から N までの数を過不足なく配置したものとする。このとき対角線の N 個のマスの合計値は、いずれも $([ア] + N) \div 2$ なる。

N が 3 の場合の魔方陣の一つを図 1 に示す。

4	9	2
3	5	7
8	1	6

図 1 3×3 の魔方陣の一つ

N が奇数の場合、魔方陣の一つを次の手順で作ることができる。 $N=3$ のときに、この手順によって 1~6 の数が配置される様子を図 2 に示す。

〔魔方陣の作り方〕

魔方陣の作り方は、次のとおりである。ここで(A)~(E)は図 2 中の該当箇所を示す。

- (1) $N \times N$ の全てのマスは何も入っていない空白の状態とする。
- (2) 最下行の中央のマスに現在位置とし、現在位置に数 1 を配置する (A)。
- (3) 現在位置の右下のマスが空白かどうか確認する。このとき、最下行の下は最上行 (B)、最右列の右は最左列 (C) とする。右下隅の右下は、左上隅 (D) である。
- (4) (3) で確認したマスが空白の場合は、そこを新しい現在位置とする。(3)で確認したマスが空白でない場合は、現在位置の上のマスを新しい現在位置とする (E)。この際、新しい現在位置が最上行よりも上になることはない。
- (5) 数を一つ増やし、現在位置にその数を配置する。
- (6) 全てのマスが埋まるまで、(3)~(5) を繰り返す。

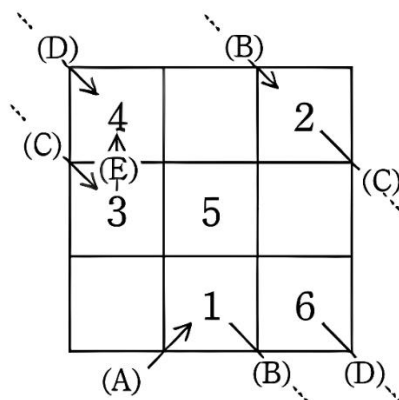


図 2 魔方陣の作り方

〔魔方陣のプログラム〕

魔方陣の数の配置を記憶する、整数型の 2 次元配列 `houjin` を用意する。配列の添字は 1 から始まる。行 y 列 x のマスは、`houjin[y][x]` で表現する。例えば、図 1 中の 1 が配置されているマスは、`houjin[3][2]` である。

数の配置に関する判定をするために、配列 `houjin` の領域を $(N+1) \times (N+1)$ の大きさで用意し、適切な初期値を設定する。 N が 3 の場合の例を図 3 に示す。数が既に配置されているかどうかを判定するために、図 3 の太枠内の各マスの初期値は 0 とする。また、現在位置の右下のマスが太枠の外であることを判定するために、4 行目のマスに `SOTO_SHITA`、4 列目のマスに `SOTO_MIGI`、行 4 列 4 のマスに `SOTO_KADO` の三つの異なる定数 (0 から N^2 までの整数以外の整数) を初期値として設定する。

		(列)			
		1	2	3	4
(行)	1	0	0	0	SOTO_MIGI
	2	0	0	0	
	3	0	0	0	
	4	SOTO_SHITA			SOTO_KADO

図 3 配列 `houjin` の初期値 (N が 3 の場合)

配列 `houjin` の初期化をする関数 `shokika`。及び数を配置する関数 `mahoujin` のプログラムを図 4 に示す。引数 N は、正の奇数 ($N \geq 3$) である。

```

function shokika(N)
  for( y を 1 から N まで 1 ずつ増やす )
    for( x を 1 から N まで 1 ずつ増やす )
      houjin[y][x] ← 0
    endfor
    イ ← SOTO_MIGI
  endfor
  for( x を 1 から N まで 1 ずつ増やす )
    ウ ← SOTO_SHITA
  endfor
  houjin[N+1][N+1] ← SOTO_KADO
endfunction

function mahoujin(N)
  y ← N
  エ
  suuji ← 1
  houjin[y][x] ← suuji

  while( suuji が オ )
    yb ← y
    xb ← x

    y ← y+1
    x ← x+1
    if( houjin[y][x] が SOTO_SHITA と等しい )
      y ← 1
    elseif( houjin[y][x] が SOTO_MIGI と等しい )
      x ← 1
    elseif( houjin[y][x] が SOTO_KADO と等しい )
      y ← 1
      x ← 1
    endif

    if( houjin[y][x] が 0 と等しくない )
      y ← カ
      x ← キ
    endif

    suuji ← suuji+1
    houjin[y][x] ← suuji

  endwhile
endfunction

```

(F)

図 4 魔方陣のプログラム

〔プログラムの判定部分の改変〕

図 4 のプログラムによるメモリ使用量の削減のために、配列 houjin の領域を NXN に縮小し、定数 SOTO_SHITA, SOTO_MIGI 及び SOTO_KADO を使わないようにするプログラムの改変を考えた。図 4 の (F) の部分を改変したプログラムを図 5 に示す。

```
y ← y+1
x ← x+1
if( y が [ ク ] よりも大きい )
    y ← [ ケ ]
endif
if( x が [ ク ] よりも大きい )
    x ← [ ケ ]
endif
```

図 5 図 4 の(F)の部分を改変したプログラム

設問 1 本文中の〔 ア 〕に入れる適切な式を答えよ。

設問 2 〔魔方陣のプログラム〕について、(1), (2)に答えよ。

(1) 図 4 中の〔 イ 〕～〔 キ 〕に入れる適切な字句を答えよ。

(2) 図 4 の関数 mahoujin を実行した場合、配列 houjin の中で一度も参照も代入もされない要素が二つ存在する。該当する配列 houjin の要素をそれぞれ答えよ。

設問 3 図 5 中の〔 ク 〕, 〔 ケ 〕に入れる適切な字句を答えよ。