SuperCon2016 認定問題と予選問題

1　魔方陣と虫食い魔方陣

N 次の魔方陣 (よく間違えられるのだが、ファンタジー作品に登場する魔法陣とは字が違うことに注意) とは、$N \times N$ マスの正方形の中に 1 から N^2 までのすべての整数を重複なく配置し、すべての縦の列、すべての横の列、ふたつの対角線に並んだ数字の合計 (それぞれ N 個の数の和) が等しくなるようにしたものである。たとえば、3 次の魔方陣は

4	9	2
3	5	7
8	1	6

である。すべての縦・横・対角線の和が 15 であることを確かめてほしい。

実は 3 次の魔方陣はほかにはこれを回転させたものと裏返したものしか存在せず、合計 8 個である。次数が増えると魔方陣の数は急激に増える。4 次・5 次の魔方陣は、回転と裏返しは同一とみなして、それぞれ 880 個と 2 億 7530 万 5224 個であることが知られている。プログラミングの練習問題として、4 次の魔方陣をすべて作ってみるのもいいだろう。6 次以上の魔方陣の数は厳密に数えられないが、統計的な手法によって 30 次までは推定されており、30 次の魔方陣は 6.6×10^{2056} 個程度と求められている。[*1]

さて、今回の SuperCon 予選問題では「虫食い魔方陣」を取り上げる。虫食い魔方陣とは魔方陣の数字のいくつかが隠されているものである。たとえば次の 4 次の魔方陣

7		2	10
14		3	11
1	9	16	
	4	13	

では、五つのマスが黒く塗りつぶされて、数字が見えなくなっている。この見えない数字を埋めて魔方陣を完成させたい。4 次の魔方陣の縦・横・対角線の和がそれぞれ 34 であること (理由は考えてみてください) と、同じ数は二度現れないことを使えば、答は

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

だとわかる。これが確かに魔方陣になっていることを確かめてみてほしい。

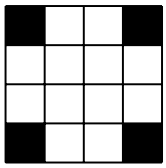
このように一部が隠された魔方陣が問題として与えられるので、隠された数字を決定して魔方陣を完成させるためのプログラムを作るのが、今回の問題である。

[*1] 興味があるかたは北島と菊池による以下の論文を見てください。
http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0125062

2 問題

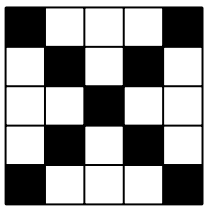
2.1 SuperCon 3 級認定問題

図のように 4 次の魔方陣の四隅が隠された虫食い魔方陣を完成させるプログラムを作れ。白いマスにはいる数字が問題として与えられるので、全体が魔方陣になるように黒いマスに入る数字を決定してほしい。



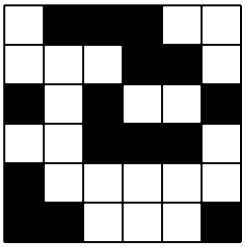
2.2 SuperCon 2 級認定問題

図のように 5 次の魔方陣の対角線すべてが隠された虫食い魔方陣を完成させるプログラムを作れ。白いマスにはいる数字が問題として与えられるので、全体が魔方陣になるように黒いマスに入る数字を決定してほしい。



2.3 SuperCon 1 級認定兼予選問題

図のように 6 次の魔方陣の中でランダムに 15 個の数字が隠された虫食い魔方陣を完成させるプログラムを作れ。どの数字が隠されるかは、各問題毎に乱数を使ってランダムに決められる。図はあくまでも虫食いの一例であり、実際の問題では別の虫食いパターンが与えられる。虫食いパターンと白いマスにはいる数字が問題として与えられるので、全体が魔方陣になるように黒いマスに入る数字を決定してほしい。なお、解はひと通りとは限らないが、複数ある場合にもその中のひとつを決定すればよい。



2.4 級の認定基準と予選の選考基準

各級とも、問題を 5 問解いてすべてに正答すれば級を認定する (1 問を解くプログラムを提出し、運営側で問題を変えて 5 回実行する)。

1 級の問題は SuperCon 予選問題を兼ねているので、本選参加希望チームは 1 級問題の解答プログラムを提出する (本選参加を希望せず、1 級認定のみを希望する場合は応募時にその旨を書くこと)。東西それぞれ、正答数の多いものから順に 10 チームを予選通過とする。正答数が同じ場合、合計の計算実行時間が短いチームから順に選ぶ。5 問で実行時間に差がつかない場合には問題数を増やすこともありうる。全問正解せずに予選通過となった場合は、本選には出場できるが 1 級は認定されない (予選通過チームは全問正解するはずと信じている)。なお、計算実行時間にはデータの入出力時間は含まれない (下で説明する入出力関数に時間測定が含まれており、入力終了時から出力直前までの時間を計測する)。

3 詳細

3.1 問題の形式

虫食い魔方陣のデータは数字の列として与えられる。その際、虫食いの部分は 0 が入っている。たとえば、最初に例に挙げた 4 次の虫食い魔方陣の入力データは

7 0 2 10 14 0 3 11 1 9 16 0 0 4 13 0

である。0 が虫食いのマスを表す。下で説明する入力関数はこのデータを標準入力から読んで、$N \times N$ の二次元配列に格納する。

解答は完成した魔方陣を同じ形式で出力する。下で説明する出力関数は二次元配列に格納された魔方陣をこの形式で標準出力に出力する。なお、ひとつの虫食い魔方陣に対して可能な魔方陣が複数ある場合には、どれかひとつだけを出力すればよい。

3.2 ヘッダーファイルと入出力関数

各級ごとにヘッダーファイルを用意してあるので、それをインクルードする。ヘッダーファイル名は 3,2,1 級それぞれ、sc3.h、sc2.h、sc1.h である。これをプログラムと同じディレクトリー (フォルダー) に置いて

```
#include "sc1.h"
```

などとする。たとえば、stdin.h と stdlib.h を使う場合、プログラムの冒頭部は

```
#include<stdio.h>
#include<stdlib.h>
#include "sc1.h"
```

のようになる。

ヘッダーファイル内では以下のように魔方陣の次数を表す定数、入出力関数、魔方陣データを収める配列、それに時間測定のための変数が定義されている。

`N`: 魔方陣の次数を表す定数 (級により、4,5,6 のいずれか)
`s[N][N]`: 魔方陣のデータを収める二次元配列
`void input(int s[][N])`: 入力関数。標準入力から魔方陣のデータを読み、配列 s に格納する。また、入力後に時刻 starttime を設定する
`void output(int s[][N])`: 出力関数。配列 s の中身を標準出力に出力する。また、出力前に時刻 endtime を設定し、配列 s の次に計算時間として endtime-starttime を出力する。
`starttime, endtime`: 時刻を格納する定数。

問題となる魔方陣のデータは input 関数により配列 s に格納される。解答は完全な魔方陣を同じ配列 s に格納し、output 関数に渡すことで出力される。なお、input 関数は必ずプログラムの最初の実行文であること。input の前に実行文を書いてはならない。

ヘッダーファイルは改変してはならない。また、入出力は必ず input と output のみで行ない、他の出力をプログラムに含めてはならない。また、starttime と endtime の値はプログラム中で変えてはならない。これらの注意に反するプログラムは失格とする。

3.3 言語と実行環境その他の注意

プログラムは ANSI 準拠の C 言語 (C99) で書く。C++ は受け付けない。また、プログラムは分割せずひとつのファイルであること (ヘッダーファイルは提出不要)。独自のヘッダーファイルやライブラリーは使用できないので、プログラム開発に統合開発環境を使用する参加者はくれぐれも注意すること。

審査の 2.5GHz の Intel Xeon CPU を搭載した LINUX マシン上で実行する。コンパイラは gcc バージョン 4.4.7 を使い、コンパイルオプションは -O のみとする (数学関数を使うプログラムについては-lm をつける)。実行時間上限は 1 問あたり 5 分 (5 問で 25 分) とし、それ以内に結果が出なかった場合は「結果なし」として扱う。

3.4 サンプル魔方陣

サンプルとして、完成した魔方陣 (虫食いではない) のデータをいくつかウェブサイトに掲載するので、プログラムのテストデータを作るのに活用してほしい (もちろん、テストデータを自分で作ってもいいのだが、6 次くらいになるといささか大変かと思う)。