情報理工学演習III　演習課題（第5回）

テーマ　　：再帰
実施日　　：2020年5月21日（木）
担当　　：有村 博紀，堀山 貴史，金森 憲太朗（TF），加井 丈志（TA）
授業連絡先：enshuiii-csit@ist.hokudai.ac.jp（演習課題の提出・質問など）
担当連絡先：kanamori@ist.hokudai.ac.jp（金森），kai@ist.hokudai.ac.jp（加井）

以下の各問に関して，それぞれプログラムを実装せよ． 実装したコードは必ず，以下に示す提出様式を守って，**本授業のメールアドレス（enshuiii-csit@ist.hokudai.ac.jp）宛にメールで提出**すること．

- メールの件名　：「[csit ex3]xxxxxxxx」，ここで xxxxxxxx は自分の学生番号．
- 添付ファイル名：第 n 回演習の問 m に対するコードのファイル名は，「n_m.c」とする．

例：学生番号02180000の学生の場合

- メール宛先　：enshuiii-csit@ist.hokudai.ac.jp
- メール件名　：[csit ex3]02180000
- 添付ファイル：5_1.c ， 5_2.c ， 5_3.c ， 5_4.c

メール本文には，学生番号，所属コース，名前，実施回（第5回演習），解いた問題番号を記載すること． 課題提出によって出席をとるため，解いている途中の状態でも構わないので，演習時間内に一度は提出すること．

実装にはC言語を用い，gcc（ver.4.8.5 以降）でコンパイルすること．

コンパイル・実行例

```
% gcc 5_1.c        // コンパイル
% ./a.out          // 実行
```

演習時間内に解き終わらなかった場合は，次回の演習時間までに提出すること．期間中であれば，何度でも再提出してよい． 提出状況等は本授業のホームページまたはELMSグループ（moodle）上に掲示予定なので，適宜確認すること．

問題番号に * が付されたものは，発展課題である．解答は必須ではないが，解ければ成績評価に加点されるので，是非挑戦してほしい．

【問1】 n 番目（$n \geq 0$）の要素 F_n が次のように再帰的に定義される数列をフィボナッチ数列という．

- $F_0 = 0$
- $F_1 = 1$
- $F_n = F_{n-2} + F_{n-1}$ （但し，$n \geq 2$）

ここで，F_n を n 番目のフィボナッチ数という．

非負整数 n を受け取り，n 番目のフィボナッチ数を出力するプログラムを 5_1.c として実装せよ．
入力は標準入力から与えられるものとする．出力の末尾には改行を入れること．

実装したプログラム 5_1.c に，標準入力から $n = 30$ を入力して実行し，結果を確認せよ．

【問2】 以下のようなパズルゲームをハノイの塔という．

横一列に並んだ3本の杭と，中央に穴の空いた互いに大きさの異なる円盤が n 枚ある．
はじめ，円盤はすべて左端の杭にあり，小さいものが上になるように順に積み重ねられている．
ここから，以下のルールに従ってすべての円盤を右端の杭に移動することを試みる．

- いずれかの杭の一番上にある円盤をひとつ選び，他の杭に移動する．これを1回の操作と数える．
- 移動先の杭に既に円盤がある場合，その円盤がいま移動しようとしている円盤よりも大きくなければならない．

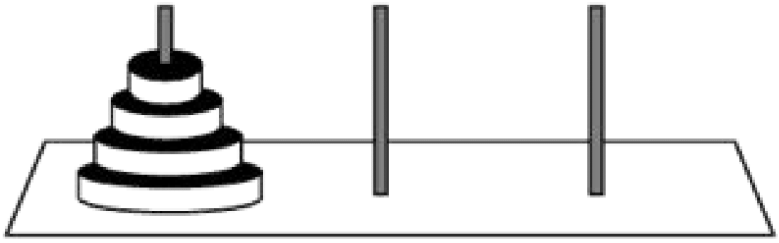

図1　4枚の円盤からなるハノイの塔．最短15回の操作で，すべての円盤を右端の杭に移動可能である．

1 以上の整数 n を受け取り，その枚数のハノイの塔を行うプログラムを 5_2.c として実装せよ．プログラムは，左端の杭に積み上げられた n 枚の円盤を，ハノイの塔のルールに従ってすべて右端の杭に移動可能か判定し，可能ならばその最短操作回数を，不可能ならば -1 を出力すること．
入力は標準入力から与えられるものとする．出力の末尾には改行を入れること．

実装したプログラム 5_2.c に $n = 25$ を入力して実行し，結果を確認せよ．

【問3*】 プログラムの繰り返し処理は，すべて再帰処理で実現可能であることが知られている．
以下の Sample 1 は，繰り返し処理（for ループ）を用いて，配列の要素を先頭から順に出力するプログラムである．このプログラムの実行結果とまったく同じ実行結果を，繰り返し処理（for ループや while ループ）を用いず，再帰処理のみで実現するプログラムを 5_3.c として実装せよ．

Sample 1

```
 1: #include <stdio.h>
 2:
 3: int main() {
 4:   int a[10] = {3, 1, 4, 1, 5, 9, 2, 6, 5, 3};
 5:   int i;
 6:   for (i = 0; i < sizeof(a)/sizeof(a[0]); i++) {
 7:         printf("a[%d]: %d\n", i, a[i]);
 8:   }
 9:
10:  return 0;
11: }
```

実装したプログラム 5_3.c を実行し，結果を確認せよ．

【問4*】 縦 12 マス，横 12 マスの地図がある．各マスには 0 または 1 が書き込まれており，0 は海を，1 は陸を表す．また，2つの 1 のマスが四近傍（斜めを含まない上下左右）のいずれかで互いに隣接しているとき，これらは地続きであるという．
この地図では，以下のいずれかの条件を満たす領域を「島」という．

- 四近傍のいずれにも 1 のマスを持たない 1 のマス
- 互いに地続きのみを辿って到達可能な 1 のマスの集合が象る領域

たとえば，以下の 図2 には5つの島がある．

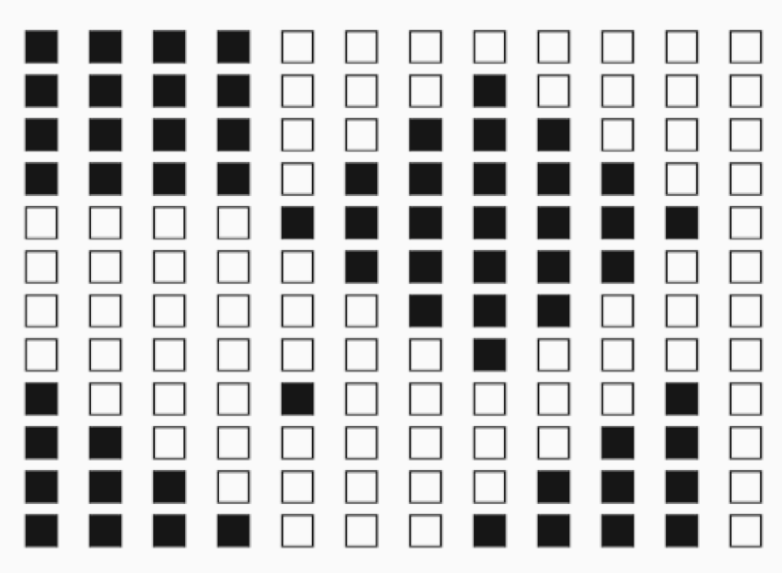

図2　地図の例．簡単のため，0 を白，1 を黒で表現している．

地図を表す2次元配列を受け取り，そこに存在する島の数を総マス数に対して線形時間で出力するプログラムを 5_4.c として実装せよ．
入力は以下の形式で標準入力から与えられるものとする．出力の末尾には改行を入れること．

入力形式

$a_{(0,0)}$ $a_{(1,0)}$ $\cdots$ $a_{(11,0)}$
$a_{(0,1)}$ $a_{(1,1)}$ $\cdots$ $a_{(11,1)}$
$\cdots$
$a_{(0,11)}$ $a_{(1,11)}$ $\cdots$ $a_{(11,11)}$　　// $a_{(x,y)}$ は 座標 (x,y) のマスの値を表す

また，以下の Sample 2 は上記入力形式のデータを受け取り，int 型の配列 grid に格納するプログラムである．

Sample 2

```
 1: #include <stdio.h>
 2: #define N 12
 3:
 4: int main() {
 5:     int i, j;
 6:     int grid[N][N];
 7:     for (i = 0; i < N; i++) {
 8:         for (j = 0; j < N; j++) {
 9:             scanf("%d", &grid[i][j]);
10:         }
11:     }
12:
13:     // 処理を実装
14:
15:     return 0;
16: }
```

実装したプログラム 5_4.c に，以下の Input 1 を入力して実行し，結果を確認せよ.

Input 1

```
0 1 0 0 0 1 1 1 1 1 0 0
1 1 0 0 1 0 0 0 0 0 1 0
0 1 0 0 1 0 0 0 0 0 0 1
0 1 0 0 0 0 0 0 0 0 0 1
0 1 0 0 0 0 0 0 0 1 1 0
0 1 0 0 0 0 1 1 1 0 0 0
0 1 0 0 0 0 0 0 0 1 0 0
0 1 0 0 0 0 0 0 0 0 1 0
0 1 0 0 0 0 0 0 0 0 0 1
0 1 0 0 1 0 0 0 0 0 0 1
0 1 0 0 1 0 0 0 0 0 1 0
1 1 1 0 0 1 1 1 1 1 0 0
```

尚，このような複数行にわたる長大な標準入力は，ファイルとして保存（たとえば，input.txt として保存）し，実行の際に以下のようにリダイレクト処理を用いてプログラムに入力する方法が便利である.

実行例

```
% ./a.out < input.txt    // 実行
```