2　仮想計算器 Nabolos

Nabolos[*1] は大昔の人が計算器を使うとしたらこんなものになるのかなという，非常に原始的な計算器のモデルです。扱う数は自然数 (0 を含めます) だけです。計算時間は非常に長くかかりますが，最新のコンピュータで計算できる関数はすべて計算できる能力があります。

2.1　ハードウェア

ハードウェアはつぎの 2 種類のものからなります。

(1)　地面に空けた穴

(2)　その中に入れる小石

穴は必要に応じていくつでも増やすことができるし，大きく広げることもできます。すなわち，穴の個数もその中に入れる石の個数も制限はありません。

それぞれの穴には，識別できるように名前がついています。たとえば異なる草や花が挿してあってその草花の名前で呼んだり，通し番号がついていてその番号で呼ぶことにします。

2.2　ソフトウェア (流れ図)

作業の手順を図で示したものを**流れ図（flow chart）**と言います。

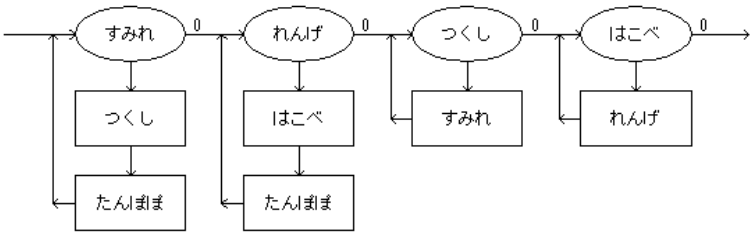
Nabolos の流れ図は，長方形と楕円と矢印で構成されます。

2.2.1　例.「足し算」の流れ図

下の図は，“すみれ” と “れんげ” の穴に入っている石を合わせた数の石を空の穴 “たんぽぽ” に入れる，すなわち

たんぽぽ ← すみれ + れんげ

を行う作業の流れ図です。



(1)　長方形は，その中に書いてある穴に石を 1 つ加えて，矢印の方向に進みます。

(2)　楕円は，その中に書いてある穴が空でないときは，その穴の石を 1 つ減らして，矢印の方向に進みます。また，その穴が空だった場合は「0」のラベルがついた矢印の方向に進みます。

[*1] 命名理由は逆から読むとわかるでしょう。

2.2.2 実行例

1.2.1 の流れ図に従って 3 + 2 を実行する様子を見てみましょう。
初めに，すみれの穴に 3 個，れんげの穴に 2 個石が入っていて，それ以外の穴は空とします。

すみれ	3	2	2	2	1	1	1	0	0	0	0	0	0	0	0	1	1	2	2	3	3	3	3
れんげ	2	2	2	2	2	2	2	2	2	2	1	1	1	0	0	0	0	0	0	0	0	1	2
たんぽぽ	0	0	0	1	1	1	2	2	2	3	3	3	4	4	4	5	5	5	5	5	5	5	5
つくし	0	0	1	1	1	2	2	2	3	3	3	3	3	3	3	2	2	1	1	0	0	0	0
はこべ	0	0	0	0	0	0	0	0	0	0	0	1	1	1	2	2	2	2	2	2	2	1	0

注意 1. この流れ図は，和を入れる穴 “たんぽぽ” と作業用の穴である “つくし” と “はこべ” が最初に空でないと正しい計算ができません。そのような場合も考慮して厳密を期すためには，これらの穴を空にする作業を先頭に追加しないといけません。

注意 2. 和を計算した後で，“すみれ” と “れんげ” の石を元通りに復元しています。復元しないで空にしたままでよければ，もっと簡単にできます。

2.2.3 例題

次の計算をする流れ図を書きなさい。結果を入れる穴と作業用に使う穴は空になっているとします。計算の元になる石が入っている穴の石は，計算後に復元するものとします。

(1) 引き算（負の数がないので，小さい数から大きい数を引いた場合の答えは 0 とする。）

$$\text{たんぽぽ} \leftarrow \begin{cases} \text{すみれ} - \text{れんげ} & (\text{すみれ} \geqq \text{れんげ}) \\ 0 & (\text{すみれ} < \text{れんげ}) \end{cases}$$

(2) 掛け算

$$\text{たんぽぽ} \leftarrow \text{すみれ} \times \text{れんげ}$$

2.3 ソフトウェア (プログラム)

作業の手順を箇条書きの文で示したものを**計画書（program）**と言います。
Nabolos のプログラムは，1 行に 1 つの**命令（command）**を書きます。命令は，流れ図の長方形，楕円，矢印に対応して 3 種類あって，それぞれ動作を表す Operation 部と，動作の対象を表す Operand 部から成ります。動作の対象は，穴の番号またはプログラムの行の番号です。

2.3.1 命令

(1) 長方形に対応する命令 Increment（増加）

```
Inc     穴の番号
```

指定した穴に石を 1 つ加えて，すぐ次の行に進みます。

(2) 楕円に対応する命令 Decrement（減少）

```
Dec      穴の番号
```

指定した穴の石を 1 つ減らして，次の次の行に進みます。ただし，指定した穴が空の場合，何もしないですぐ次の行に進みます。

(3) 矢印に対応する命令 Jump（移動）

```
Jmp      行の番号
```

指定した行に進みます。

2.3.2 例.「足し算」のプログラム

1.2.1 の流れ図に対応するプログラムです。ただし，5 つの穴 “すみれ”，“れんげ”，“たんぽぽ”，“つくし”，“はこべ” の番号を順に，1,2,3,4,5 とします。

```
 1     Dec     1
 2     Jmp     6
 3     Inc     4
 4     Inc     3
 5     Jmp     1
 6     Dec     2
 7     Jmp     11
 8     Inc     5
 9     Inc     3
10     Jmp     6
11     Dec     4
12     Jmp     15
13     Inc     1
14     Jmp     11
15     Dec     5
16     Jmp     19
17     Inc     2
18     Jmp     15
19
```

注意 3. 行番号は，参考のために書いたもので，プログラムには含まれません。

2.3.3 相対ジャンプ

Jmp 命令は次に実行する命令の行番号を指定します。プログラムを作るときに，行を追加したり削除したりすると，その度に行番号を書き換えないといけません。行番号を相対的に指定する，すなわちその命令がある行から上または下へ何行目と指定すると，その間に行を追加または削除しない限り書き換えないですみます。

```
Jmp     n       / n行目に進む（絶対指定）
Jmp     .+n     / n行下に進む（相対指定）
Jmp     .-n     / n行上に進む（相対指定）
```

2.3.4　例.「足し算」のプログラム

1.3.2 のプログラムを相対 Jmp を使って書き換えると次のようになります。

```
 1    Dec    1
 2    Jmp    .+4
 3    Inc    4
 4    Inc    3
 5    Jmp    .-4
 6    Dec    2
 7    Jmp    .+4
 8    Inc    5
 9    Inc    3
10    Jmp    .-4
11    Dec    4
12    Jmp    .+3
13    Inc    1
14    Jmp    .-3
15    Dec    5
16    Jmp    .+3
17    Inc    2
18    Jmp    .-3
19
```

よい副作用として，1 行〜5 行と 6 行〜10 行，および 11 行〜14 行と 15 行〜18 行が，穴の番号が異なるだけで同じ操作をしていることがわかりやすくなりました。

2.3.5　例題

1.2.3 の「引き算」と「掛け算」の流れ図をプログラムに書き直しなさい。

2.4 問題

以下の関数を計算する Nabolos プログラムを作成しなさい。ただし，x, y, z, w はそれぞれ Memory の 1,2,3,4 番目とします。また，実行開始するとき，入力変数以外の Memory は空になっているものとします。

(1) 累乗 Power.nab

$$z \leftarrow x^y$$

[ヒント]

$$x^y = \begin{cases} 1 & (y = 0) \\ x^{y-1} \times x & (y > 0) \end{cases}$$

(2) 割り算 DivMod.nab

$$z \leftarrow x \text{ div } y \quad (\text{商})$$
$$w \leftarrow x \bmod y \quad (\text{余り})$$

ただし，$y = 0$ のときは値なし，すなわち停止しないものとします。

(3) 最大公約数 Gcd.nab

$$z \leftarrow \gcd(x, y) \quad (\text{最大公約数})$$

[ヒント]　ユークリッドの互除法

$$\gcd(x, y) = \begin{cases} x & (y = 0) \\ \gcd(y, x \bmod y) & (y \neq 0) \end{cases}$$

(4) 素数判定 Prime.nab

$$y \leftarrow \begin{cases} 1 & (x \text{ が素数}) \\ 2 & (x \text{ が非素数}) \end{cases}$$

[ヒント]　$x > 2$ について，

x を $p = 2, 3, \ldots$ で割っていって
$x \bmod p = 0$　になれば　x は合成数
$x \text{ div } p < p$　になれば　x は素数

(5) 大数生成 Big.nab
Memory がすべて 0 の状態から始めて，Memory[1] になるべく大きな数を生成して停止する。
20 行（コメントや空の行を除く）で，いくつまで生成できるでしょうか。