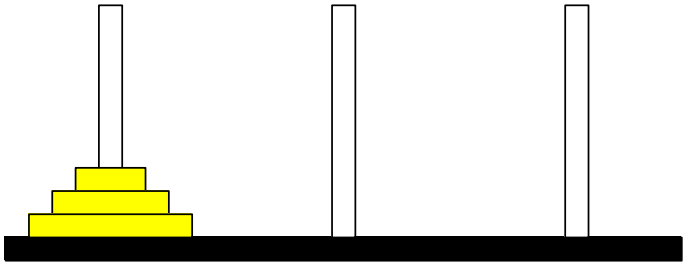
ハノイの塔

3本の杭(peg)があり，最初の杭には，おおきな円盤(ring)から小さな円盤が下から順に積まれて塔のようになっている．適宜3本の杭に円盤を差しかえて，円盤を最初の杭から最後の杭へすべて移し換えなさい．ただし，一度に一つの円盤しか動かせない．また，小さい円盤の上に大きな円盤を置いてはいけない．



課題の背景

ベナレスにあるバラモン教の大寺院は世界の中心と記されており，そこに，杭が3本立った真鍮製の基盤がある．神はその一本に64枚の純金の円盤をはめた．僧侶たちは昼夜の別なくそれを別の杭に移し替える作業に専念している．そして移し替えが完了したとき，寺院もバラモン僧たちも崩壊し，この世が塵に帰るという．
『ハノイの塔』は1883年にフランスの数学者E.リュカ（Edouard Lucas）が考えたゲームです．リュカがこの物語を創作したのか，はたまたこの物語から発案したのかは不明(R. Douglas Hofstadter. Metamagical themas. Scientific American, 248(2):16-22, March 1983.).

表示スクリプト

```
> with(plots):with(plottools):
  base:=rectangle([-4,0],[25,-1],color=black):
  peg1:=rectangle([0,0],[1,10],color=white):
  peg2:=rectangle([10,0],[11,10],color=white):
  peg3:=rectangle([20,0],[21,10],color=white):
  ring1:=rectangle([-3,0],[4,1],color=yellow):
  ring2:=rectangle([-2,1],[3,2],color=yellow):
  ring3:=rectangle([-1,2],[2,3],color=yellow):
  display([ring1,ring2,ring3,peg1,peg2,peg3,base],axes=none,
  scaling=constrained);
```

ヒント

課題:MakeHanoi:=proc(n)

円盤の枚数nを受け取って，3本の杭(peg)A,B,Cを表わす配列を初期化する関数を作れ．

1. globalでA,B,Cをとり，A:=Array(0..n);などとして杭を定義．
2. 各杭A,B,Cの第0要素には杭に刺さっている円盤の枚数を入れる．
3. A,B,Cのその他の各要素には，それぞれの高さ(indexで指定)に刺さっている円盤のサイズをいれる．
4. 初期値は，A[1]:=n;A[2]:=n-1;...A[A[0]]:=1;その他は0となる．

課題:PrintHanoi:=proc()

3本の杭A,B,Cに刺さっている円盤のサイズと位置を

```
A:  3  2  1
B:
C:
-----
```

と表示する関数を作れ．

1. A,B,Cはglobalで指定せよ．
2. できたらPrintHanoi()をMakeHanoi関数の最後に加えておけ．

課題:MovePlate:=proc(Pfrom,Pto)

杭Pfromの一番上に載っている円盤一枚を杭Ptoへ動かす操作を記述せよ．

1. 動作，表示を
   ```
   > MakeHanoi(2);
     MovePlate(A,B);
     PrintHanoi();
   ```
   として確認せよ．
2. Pfromの杭の一番上のplateをPtoの一番上にのせ，Pfrom[0],Pto[0]を調節する．
3. できたらPrintHanoi()をMovePlate関数の最後に加えておけ．

課題:Hanoi2関数

n=2の場合の円盤の動きを制御するHanoi2:=proc(Pfrom,Pto,Pwork)を作れ．ここで，Pworkは作業用の杭で，使い方は以下の表示を参照せよ．

1. 杭Aにある円盤を，杭Bを作業用の杭として使って，杭Cに移す場合
   ```
   > MakeHanoi(2);
     Hanoi2(A,C,B);
   ```
   とする．結果は以下のようになる．

```
A:  2  1
```

```
B:
C:
-----
A:  2
B:  1
C:
-----
A:
B:  1
C:  2
-----
A:
B:
C:  2  1
-----
```

課題:Hanoi3関数

n=3の場合の動きを予測し，Hanoi3:=proc(Pfrom,Pto,Pwork)を作れ．

1. Hanoi2を内部で使え．
2. 3枚の円盤のうち上部のサイズ1,2の2枚をHanoi2を使って杭Aから杭Bへ移す．次にサイズ3の円盤を杭Cに移す．最後に，杭Bにある円盤をHanoi2を使って杭Cに移せばよい．

課題:Hanoi:=proc(i,Pfrom,Pto,Pwork)

先程のHanoi3を参照して，i枚の場合のHanoi関数を作れ．

1. 一般化した場合は，i枚の円盤うち，i-1枚の円盤を杭Pfromから杭Ptoを使って杭Pworkに移す．次にサイズiの円盤を杭Ptoに移す．次に杭Pworkにあるi-1枚の円盤を杭Pfromを使って杭Ptoに移す．
2. iを引数に加えて，再帰的にHanoi関数を使うことによってハノイの塔の移動が自動的に達成される．
3. 最後をどうするかはいい加減でいい．例えば，
   i=2まできた時にはHanoi2(Pfrom, Pto, Pwork)を使う．
   あるいはi>0ではこの操作を続け，i=0では何もしない．
4. 次のようにすれば動作するはず．やってみ．

```
> n:=4;
  Makeanoi(n);
  Hanoi(n,A,C,B);
```

課題:世界の終わりはいつ？

一枚の金貨を別の杭に移すのに1秒かかるとすると，バラモン僧たちが64枚の金貨を移すのに何年かかるか？

1. 移動の回数を記録するglobal変数とその操作をMovePlateに加えて，n:=6;ぐらいまでとって，金貨の枚数nと移動回数との関係を予測せよ．
2. ちなみに宇宙年齢は137億歳(1.37*10^(11))と計測されています．後

何年？？

list版

Arrayを使うのが，配列のindexが0から始まる場合には自然です．しかし，どうしてもリスト[]を使いたい場合は以下のようにして作ることが可能．リストを作る場合の問題は，

1．円盤を指すindexがずれる．
2．procに引数として読み込んだ変数を直接変更することができない．

2はprocへの引数が値渡しなのか，変数渡しのなのかに関わる問題．これを回避するには，evaln(list)と明示すればよい．

```
> MakeHanoi:=proc(n)
  global A,B,C;
  A:=[n,seq(n-i+1,i=1..n)];
  B:=[0,seq(0,i=1..n)];
  C:=[0,seq(0,i=1..n)];
  PrintHanoi();
  end proc;
```

$$MakeHanoi := \mathbf{proc}(n) \qquad (2.1)$$

```
    global A, B, C;
    A := [n, seq(n - i + 1, i = 1 ..n) ];
    B := [0, seq(0, i = 1 ..n) ];
    C := [0, seq(0, i = 1 ..n) ];
    PrintHanoi( )
end proc
```

```
> PrintHanoi:=proc()
  global A,B,C;
  local i;
    printf("A:");
    for i from 1 to A[1] do
      printf("%3d",A[i+1]);
    end do;
    printf("\n");
    printf("B:");
    for i from 1 to B[1] do
      printf("%3d",B[i+1]);
    end do;
    printf("\n");
    printf("C:");
    for i from 1 to C[1] do
      printf("%3d",C[i+1]);
    end do;
    printf("\n");
    for i from 1 to 4 do
      printf("---");
    end do;
    printf("\n");
  end proc:
> MakeHanoi(3);

A:  3  2  1
B:
C:
------------

> MovePlate:=proc(Pfrom::evaln(list),Pto::evaln(list))
  local m;
```

```
   m:=Pfrom[Pfrom[1]+1];
   Pfrom[Pfrom[1]+1]:=0;
   Pfrom[1]:=Pfrom[1]-1;


   Pto[1]:=Pto[1]+1;
   Pto[Pto[1]+1]:=m;


   PrintHanoi();

   return;
   end proc:
> MakeHanoi(3);
   MovePlate(A,B);
A:  3  2  1
B:
C:
------------
A:  3  2
B:  1
C:
------------

> Hanoi:=proc(i, Pfrom::evaln(list),
   Pto::evaln(list),
   Pwork::evaln(list))
   if i=0 then
     return;
   end if;
   Hanoi(i-1,Pfrom,Pwork,Pto);
   MovePlate(Pfrom,Pto);
   Hanoi(i-1,Pwork, Pto, Pfrom);
   end proc:
> n:=3:
   MakeHanoi(n);
   Hanoi(n,A,C,B);
A:  3  2  1
B:
C:
------------
A:  3  2
B:
C:  1
------------
A:  3
B:  2
C:  1
------------
A:  3
B:  2  1
C:
------------
A:
B:  2  1
C:  3
------------
A:  1
B:  2
C:  3
------------
A:  1
B:
C:  3  2
------------
A:
B:
C:  3  2  1
------------
```