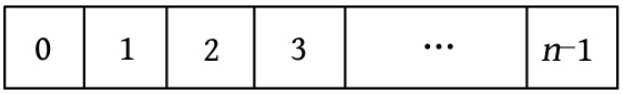
区块世界

(block.cpp/.c)

限制：1S　256MB

题目描述：

在早期的人工智能规划和机器人研究中使用了一个区块世界，在这个世界中，机器人手臂执行涉及区块操作的任务。问题是要解析一系列命令，这些命令指导机器人手臂如何操作平板上的块。最初，有 n 个区块（编号为 0～n−1），对于所有 0≤i<n−1 的情况，区块 b_i 与区块 b_{i+1} 相邻，如下图所示。



用于操纵块的有效命令如下。

- move a onto b：把 a 和 b 上方的块全部放回初始位置，然后把 a 放到 b 上方。
- move a over b：把 a 上方的块全部放回初始位置，然后把 a 放到 b 所在块堆的最上方。
- pile a onto b：把 b 上方的块全部放回初始位置，然后把 a 和 a 上方所有的块整体放到 b 上方。
- pile a over b：把 a 和 a 上方所有的块整体放到 b 所在块堆的最上方。
- quit：结束标志。

任何 a=b 或 a 和 b 在同一块堆中的命令都是非法命令。所有非法命令都应被忽略。

输入：(block.in)

输入的第 1 行为整数 n（0<n<25），表示区块世界中的块数。后面是一系列块命令，每行一个命令。在遇到 quit 命令之前，程序应该处理所有命令。所有命令都将采用上面指定的格式，不会有语法错误的命令。

输出：(block.out)

输出应该包含区块世界的最终状态。每一个区块 i（0≤i<n）后面都有一个冒号。如果上面至少有一个块，则冒号后面必须跟一个空格，后面跟一个显示在该位置的块列表，每个块号与其他块号之间用空格隔开。不要在行末加空格。

输入样例	输出样例
10 move 9 onto 1 move 8 over 1 move 7 over 1 move 6 over 1 pile 8 over 6 pile 8 over 5 move 2 over 1 move 4 over 9 quit	0: 0 1: 1 9 2 4 2: 3: 3 4: 5: 5 8 7 6 6: 7: 8: 9:

题解

初始时从左到右有 n（0<n<25）个块，编号为 0～n-1，要求实现一些操作。通过这些操作可以归纳总结出以下规律。

- move：将 a 上方的块全部放回初始位置。
- onto：将 b 上方的块全部放回初始位置。
- 公共操作：将 a 和 a 上方所有的块整体放到 b 所在块堆的最上方。

而实际上，前两种可以算一个操作：将 a（或 b）上方的块全部放回初始位置，简称归位。将 a 和 a 上面所有的块整体放到 b 所在块堆的最上方，简称移动。

只需通过判断执行归位和移动操作就可以了。

1. 算法设计

（1）读取操作命令 s1，如果 s1="quit"，则结束；否则执行下两步；

（2）读入操作命令 a s2 b，如果 s2="move"，则 a 归位；如果 s2="onto"，则 b 归位；

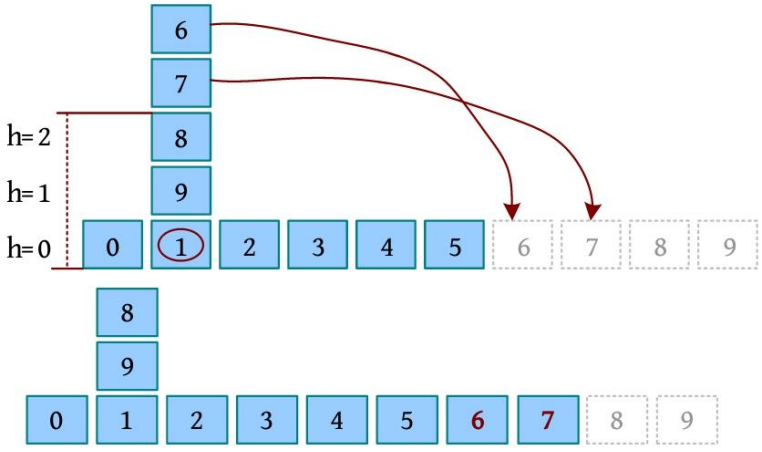
（3）执行移动操作，即将 a 和 a 上方所有的块整体放到 b 所在块堆的最上方。

那么如何执行归位和移动操作呢？

1）归位

要想使 a 上方的所有块归位，则首先要找到 a 所在的块堆，并知道 a 在块堆中的位置（高度），然后才能将 a 上方的所有块归位。

例如，块堆如下图所示，将 8 上方所有的块归位。首先查找到 8 所在的块堆为 1，8 所在块堆的高度为 2，然后将 1 号块堆高度大于 2 的所有块放回原来的位置。



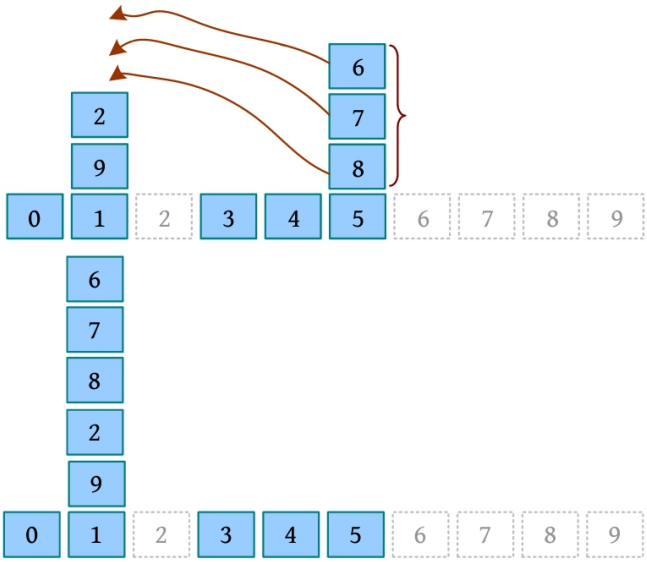
算法代码：

```
void goback(int p,int h){//将p块堆高度大于h的所有块归位
   for(int i=h+1;i<block[p].size();i++){
      int k=block[p][i];
      block[k].push_back(k);
   }
   block[p].resize(h+1);//重置大小
}
```

2）移动

要想将 a 和 a 上方所有的块整体放到 b 所在块堆的最上方，则首先要找到 a 和 b 所在的块堆，如果 a、b 所在的块堆一样，则什么都不做。否则，将 a 块堆中高度大于或等于 h（a 的高度）的所有块移动到 b 所在块堆的上方。

例如，块堆如下图所示，将 8 和 8 上方所有的块整体放到 9 所在块堆的最上方。首先查找到 8 所在的块堆为 5 号，9 所在的块堆为 1 号，8 所在块堆的高度为 1，然后将 5 号块堆高度大于或等于 1 的所有块放到 1 号块堆的上方，如下图所示。



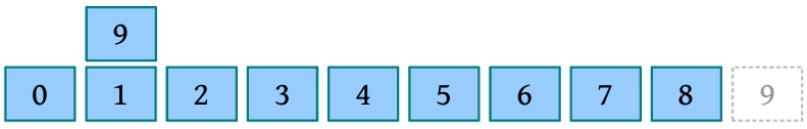
算法代码：

```
void moveall(int p,int h,int q){//将p块堆高度大于或等于h的所有块都移动到q块堆的上方
   for(int i=h;i<block[p].size();i++){
      int k=block[p][i];
      block[q].push_back(k);
   }
   block[p].resize(h);//重置大小
}
```

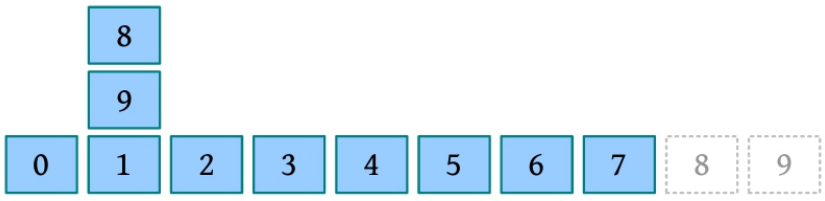
2. 完美图解

以输入样例为例，有 10 个块，初始时各就其位，如下图所示。

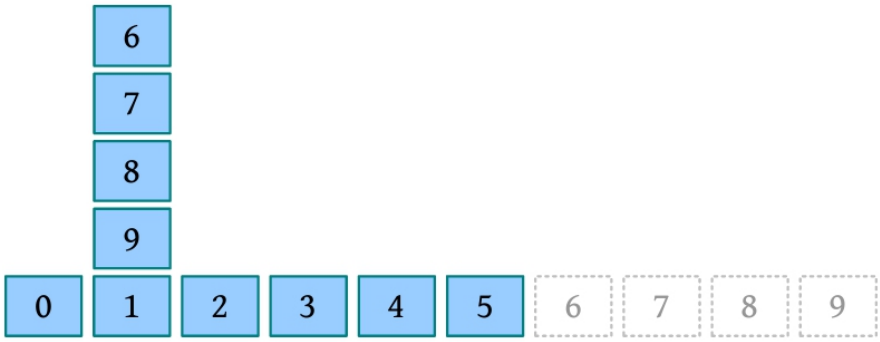
（1） move 9 onto 1：将 9 和 1 上方的块全部放回初始位置，然后把 9 放到 1 的上方。



（2） move 8 over 1：将 8 上方的块全部放回初始位置，然后把 8 放到 1 的上方。

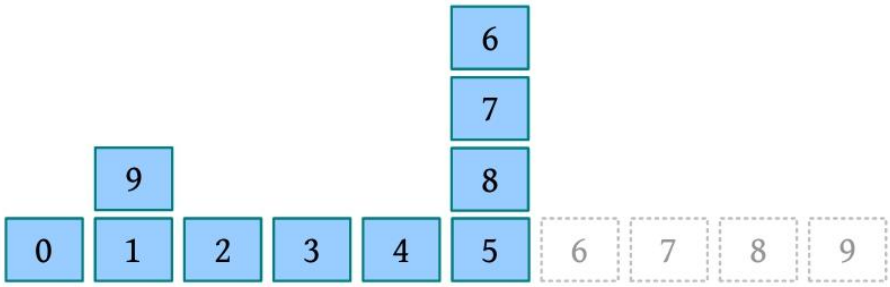


（3） move 7 over 1：将 7 上方的块全部放回初始位置，然后把 7 放到 1 的上方。move 6 over 1：将 6 上方的块全部放回初始位置，然后把 6 放到 1 的上方。

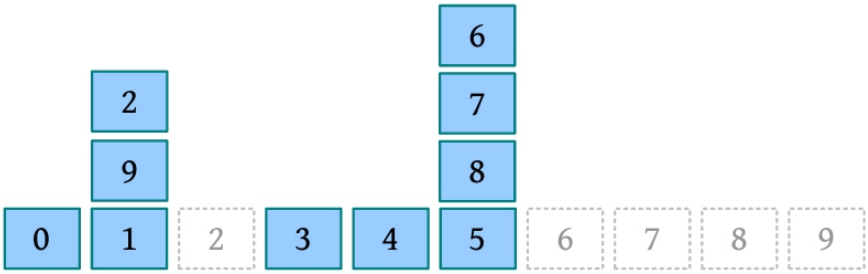


（4）pile 8 over 6：将 8 和 8 上方所有的块整体放到 6 所在块堆的最上方；此时 8 和 6 在同一块堆中，什么也不做。

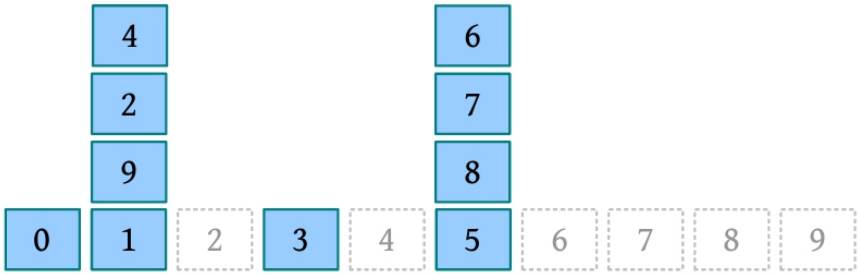
（5）pile 8 over 5：将 8 和 8 上方所有的块整体放到 5 所在块堆的最上方，即将 8、7、6 一起放到 5 所在块堆的上方。



（6）move 2 over 1：将 2 上方的块全部放回初始位置，将 2 放到 1 所在块堆的最上方。



（7）move 4 over 9：将 4 上方的块全部放回初始位置，将 4 放到 9 所在块堆的最上方。



（8）quit：结束。

（9）从左到右、从下到上输出每个位置的块编号。

3. 算法实现(**block.cpp**)

因为每一个块堆的长度都发生了变化，因此可以使用变长数组 vector，即对每个块堆都用一个 vector 存储。块堆的个数为 n（0<n<25），定义一个长度比 25 稍大的 vector 数组即可。

```
vector<int>block[30];
void init(){
   cin>>n;
   for(int i=0;i<n;i++)
      block[i].push_back(i);
}

void loc(int x,int &p,int &h){//找位置
   for(int i=0;i<n;i++)
      for(int j=0;j<block[i].size();j++){
         if(block[i][j]==x){
            p=i;
            h=j;
         }
      }
}
void goback(int p,int h){//将p块堆高度大于h的所有块归位
   for(int i=h+1;i<block[p].size();i++){
      int k=block[p][i];
      block[k].push_back(k);
   }
   block[p].resize(h+1);//重置大小
}

void moveall(int p,int h,int q){//将p块堆高度大于或等于h的所有块都移动到q块堆的上方
   for(int i=h;i<block[p].size();i++){
      int k=block[p][i];
      block[q].push_back(k);
   }
   block[p].resize(h);//重置大小
}
void solve(){
   int a,b;
   string s1,s2;
   while(cin>>s1){
      if(s1=="quit")
         break;
      cin>>a>>s2>>b;
      int ap=0,ah=0,bp=0,bh=0;
      loc(a,ap,ah);
      loc(b,bp,bh);
      if(ap==bp)
         continue;
      if(s1=="move")//a归位
         goback(ap,ah);
      if(s2=="onto")//b归位
         goback(bp,bh);
      moveall(ap,ah,bp);
   }
```

```
void print(){
   for(int i=0;i<n;i++){
      cout<<i<<":";
      for(int j=0;j<block[i].size();j++)
         cout<<" "<<block[i][j];
      cout<<endl;
   }
}
```