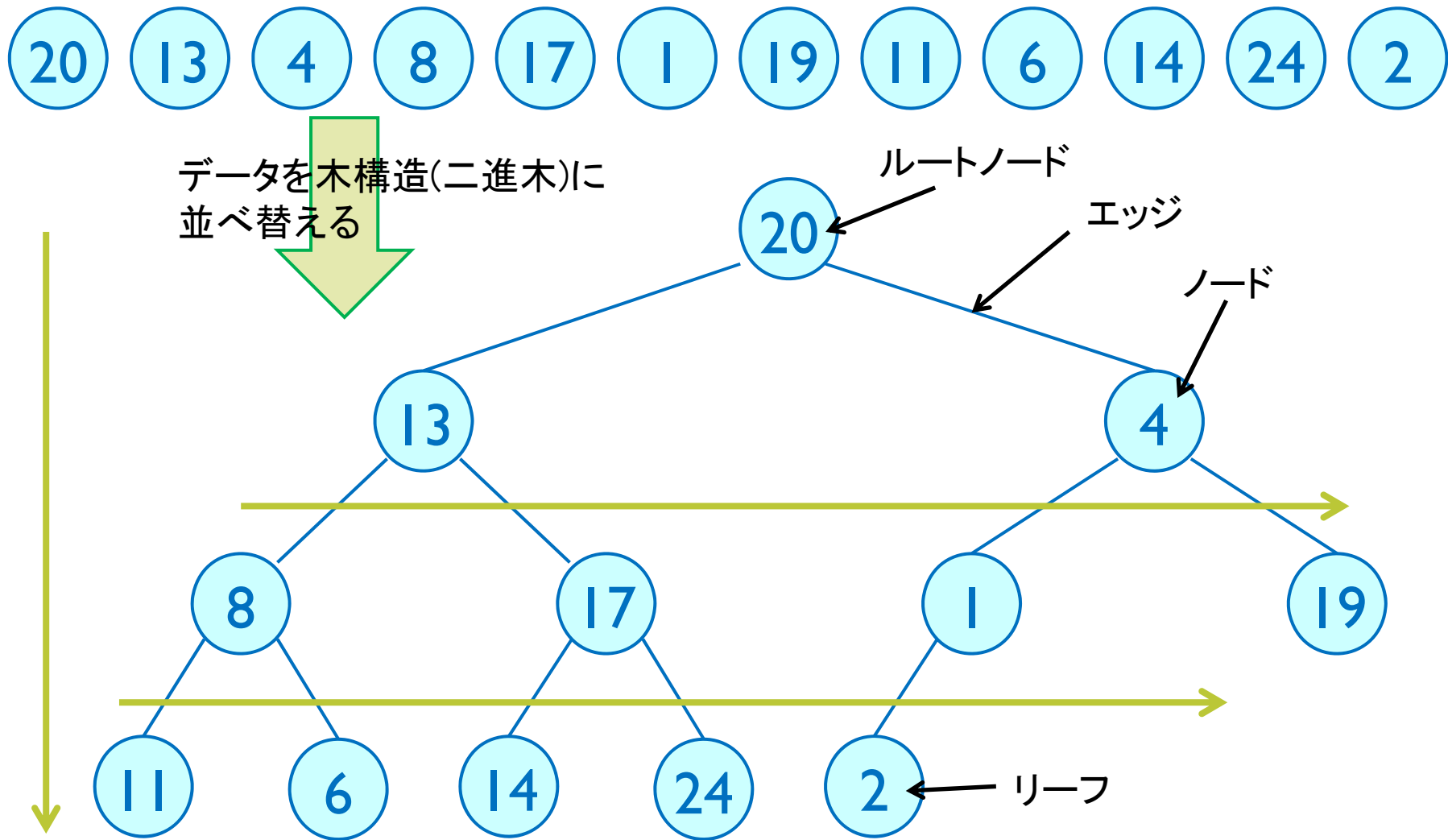


6. ヒープソート

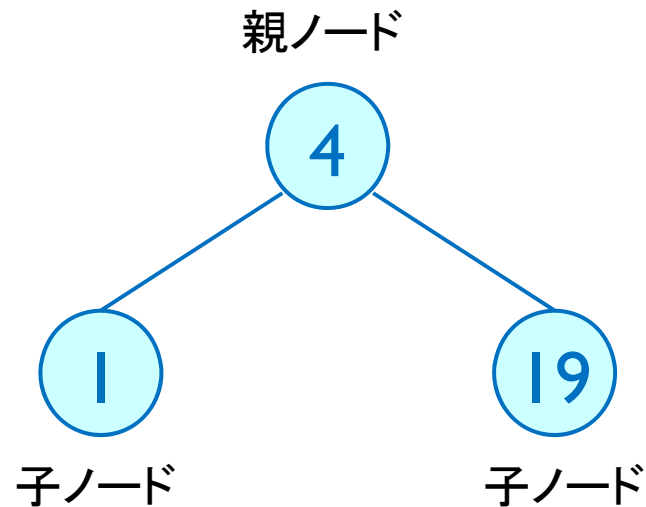
実践プログラミング I
情報工学科 鈴木雅人

ヒープソートのアルゴリズム

(1) 木構造に並べる (実際には何もしない)



ヒープソートのアルゴリズム



※ ノードの番号

親ノードの番号 = k



子ノードの番号 = $2k+1, 2k+2$

(1) ヒープ条件

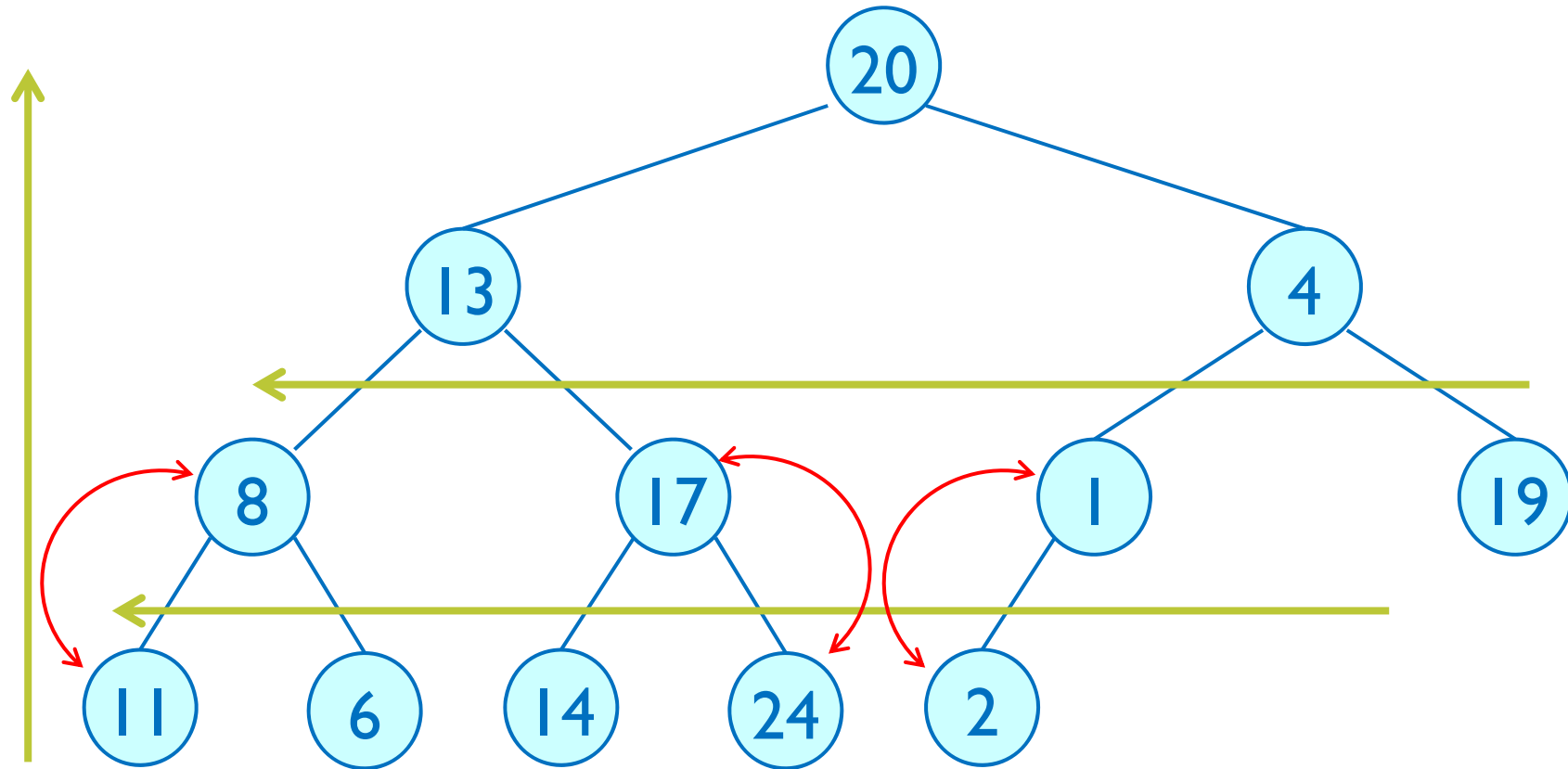
親ノードの値 \geq 全子ノードの値

(2) ヒープにするための手順

```
if( d[k] < d[2*k+1] ) {  
    /* d[k]とd[2*k+1]とを交換 */  
}  
if( d[k] < d[2*k+2] ) {  
    /* d[k]とd[2*k+2]とを交換*/  
}
```

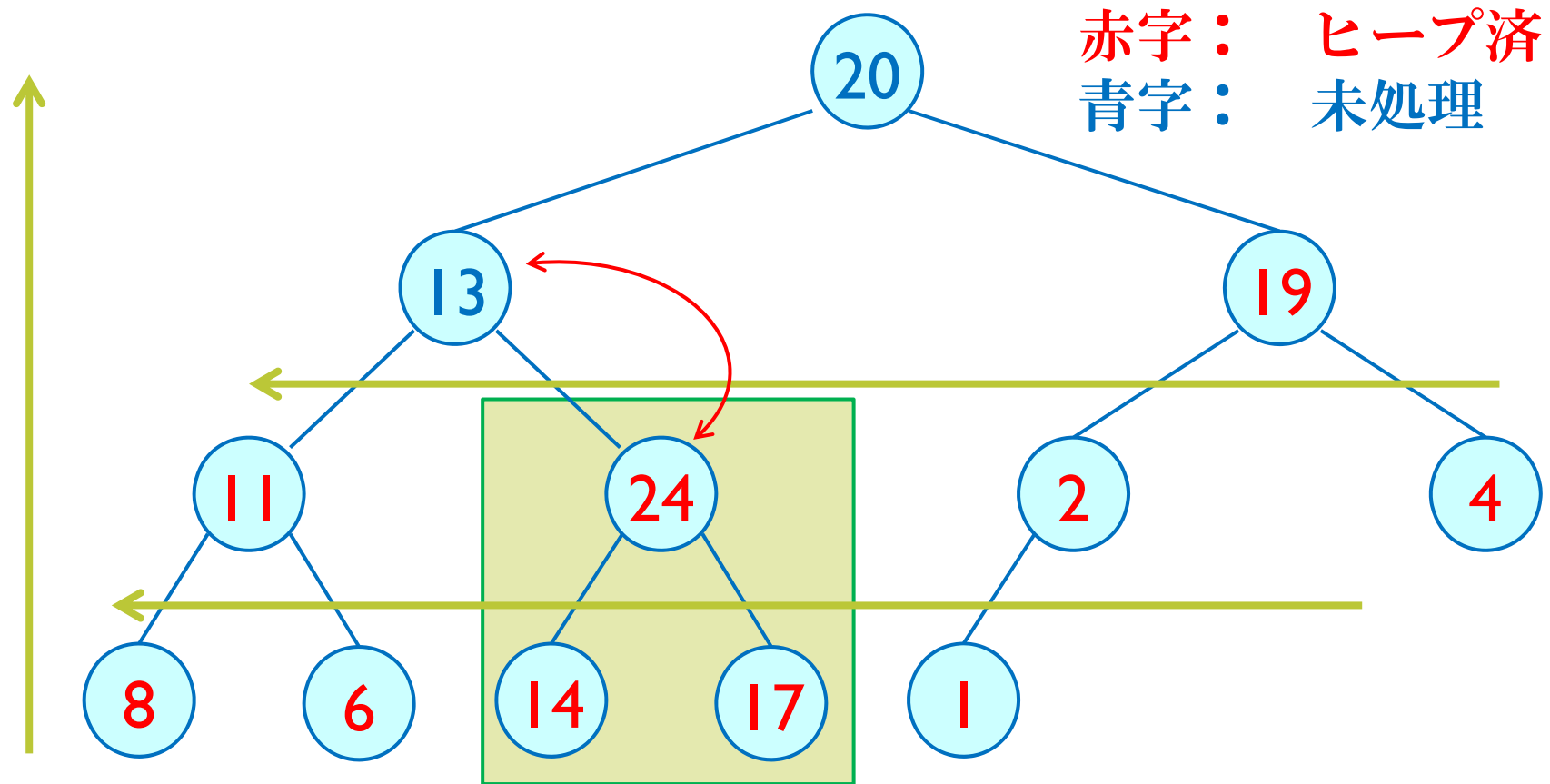
ヒープソートのアルゴリズム

(2) 木全体をヒープにする



ヒープソートのアルゴリズム

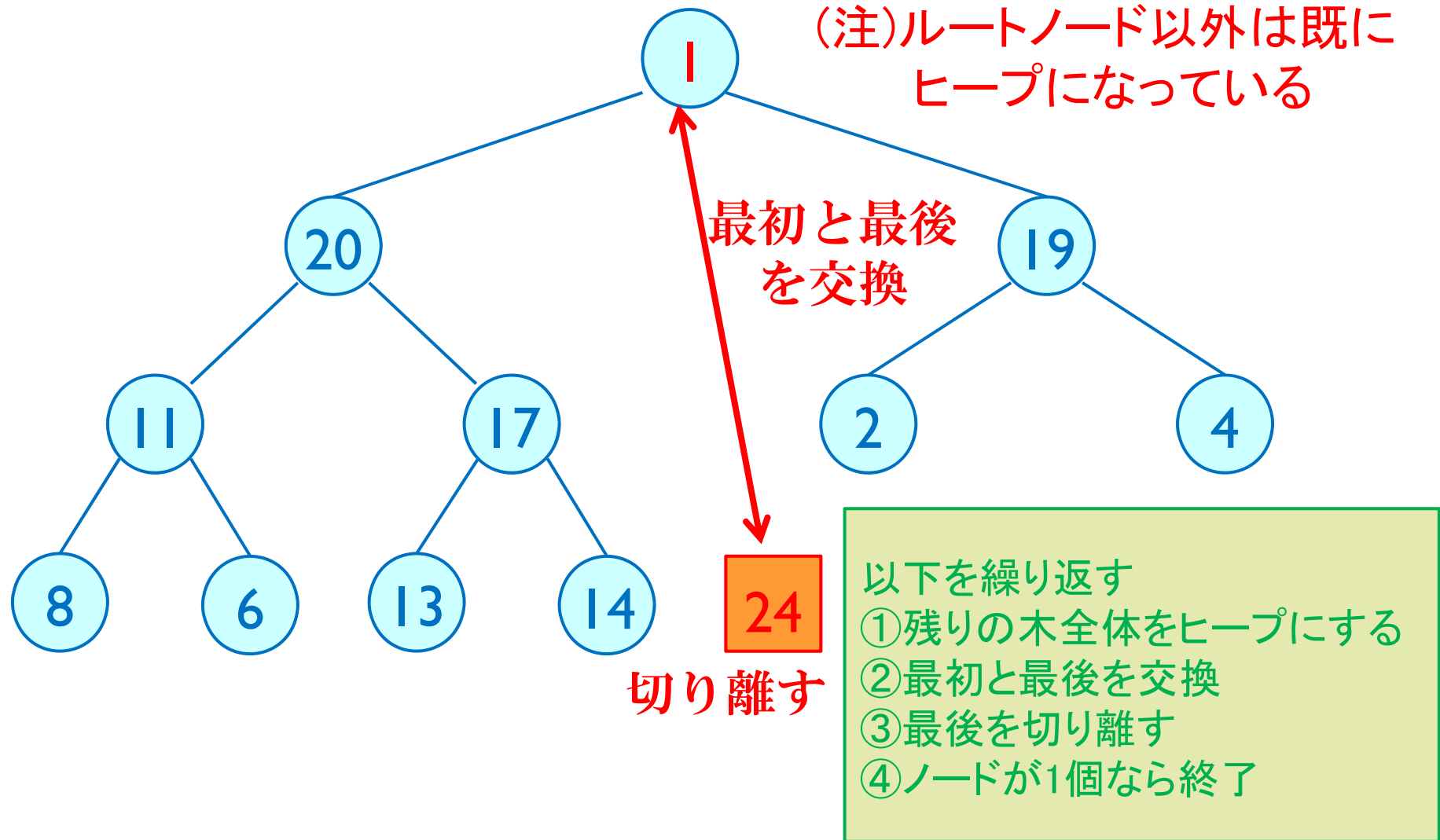
(2) の途中経過



ヒープの関係が崩れる!!
(再帰処理が必要)

ヒープソートのアルゴリズム

(3) 最大値の確定



ヒープソートのアルゴリズム

- ① データ全体をヒープにする
- ② 最初のノードと最後のノードを交換し、最後のノードの場所を確定する
- ③ 残りのノードの数が1なら終了。そうでない場合は①へ

注：最初にデータ全体をヒープにしてしまえば、2回目以降はルートノードの部分のみヒープ条件を確認すれば良い。

注：ヒープにする作業は再帰的に行う必要がある。



プログラムの設計方法 (heap関数)

`heap(int d[], int n, int idx)`

- ▶ `d[]`はデータを格納した配列
- ▶ `n`は現在ソートの対象としているデータ数
最初は`n`はデータ数を表すが1ずつ減って最後は2になる
- ▶ `idx`はヒープにしたい場所の親ノードの番号

注:`idx`番目の部分をヒープにしたときに, $idx*2+1$ 番目,
 $idx*2+2$ 番目がヒープになっているかどうか調べ, ヒープの関係が崩れた場合は再帰呼出しでヒープの関係を修復する

注: 子ノードがない場合の例外処理が必要である



プログラムの設計方法 (main関数)

```
for( k = n-1 ; k >= 0 ; k-- ) {  
    heap( d, n, k ) ;  
}  
while( n > 1 ) {  
    tmp = d[0] ;  
    d[0] = d[n-1] ;  
    d[n-1] = tmp ;  
    n-- ;  
    heap( d, n, 0 ) ;  
}
```



【課題6-1】（発展課題）

ファイルに書き込まれている100万件（または1000万件）のデータを読み込み，ヒープソートを用いてそれらを小さい順に並べ替え，結果を画面に出力するプログラムを作成しなさい。

サンプルデータは第5回課題のものを使うこと。

