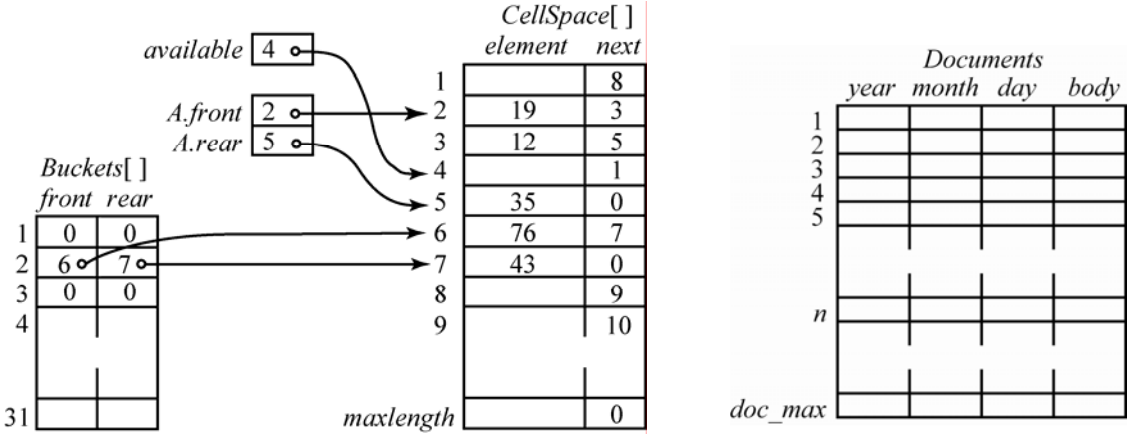
# アルゴリズムとデータ構造1　　演習問題 (6)　　回答

カーソルを用いて待ち行列（キュー）を実現するための CCELL 型の配列 *CellSpace*[1..*maxlength*] が，次のように定義されている．

```
typedef     int          cursor ;          /*  カーソルはポインタの役目をする整数  */
typedef  struct {
            elementtype  element ;         /*  elementtype 型と定義された要素を入れる欄  */
            cursor       next ;            /*  次のセルを指すカーソル  */
        } CCELL ;
typedef  struct {
            cursor       front ;           /*  キューの先頭のセルを指すカーソル  */
            cursor       rear ;            /*  キューの最後のセルを指すカーソル  */
        } cQUEUE ;

CCELL       CellSpace[1..maxlength] ;      /*  キューを連結リストで覚えるための配列  */
cursor      available ;                    /*  使用可能なセルのリストの先頭のセルを指すカーソル  */

/*  CellSpace[・] 内の現在使用されていないセルは，next 欄を用いて連結リストのように覚えられている  */
/*  その連結リストの先頭のセルをカーソル available が示す  */
/*  available が示す連結リストの最後のセルの next 欄には，NULL を意味する 0 が入っている  */
/*  CellSpace および available は大域的（グローバル）変数とする  */
/*  この問題では，CellSpace[・] 内に作られるCLIST型の連結リストはヘッダセルを持たないものとする  */
/*  キューQ が要素を持たないとき，Q.front および Q.rear には共に 0 が入っている  */
/*  キューQ の欄 Q.rear が指す Q の最後のセルの next 欄には，NULL を意味する 0 が入っている  */
```

このキューに覚える elementtype 型の要素もカーソルで，日付の付いた書類データを指している．この書類データは，DOCCELL 型の配列 *Documents*[・] に入っている．この配列は大域的（グローバル）変数である．

```
typedef     cursor       elementtype ;     /*   DOCCELL 型の配列の要素を指すカーソル  */

typedef  struct {
            int          year ;            /*  年を表す整数の欄  */
            int          month ;           /*  月を表す整数の欄  */
            int          day ;             /*  日を表す整数の欄  */
            bodytype     body ;            /*  bodytype 型のデータを入れる欄  */
        } DOCCELL ;

DOCCELL     Documents[1..doc_max] ;        /*  書類データを蓄えている配列  */
```



```
void   Radix_Sort(   cQUEUE↑  ptr_A   )
     /*  CellSpace[・] 内のキュー  A = ↑ptr_A  に入っている書類データを月日の昇順に並べ替える  */
     /*  キュー  A には, 年 (year) が全て同一の書類データへのカーソルが入っている  */
{
        cQUEUE       Buckets[1..31] ;                    /*  日および月に関するバケットソートを行うためのバケツ  */

        if (   (↑ptr_A).front  = 0  ) {
             "並べ替えるセルが無いと出力する"   ;
        }
        else {
             Bucket_Sort(   ptr_A,   31,   Buckets   ) ;          /*  日に関するバケットソートを行う  */
             Bucket_Sort(   ptr_A,   12,   Buckets   ) ;          /*  月に関するバケットソートを行う  */
        }
} /*   Radix_Sort   */
```

```
void   Bucket_Sort(   cQUEUE↑  ptr_A ,   int   b_size,   cQUEUE   Buckets[1..b_size]   )
     /*  CellSpace[・] 内のキュー  A = ↑ptr_A  に入っている書類データを月あるいは日の昇順に並べ替える  */
     /*  b_size = 31 ならば書類データの日の昇順に, b_size = 12 ならば月の昇順に並べ替える  */
{
     cursor     doc_no ;                                  /*  書類番号を示すカーソルで局所変数  */
     int     b ;                                          /*  バケツを示すために導入した局所変数  */

     for (   1 ≤ b ≤ b_size  なる各 b   ) {
             Buckets[b].front  :=  0 ;                    /*  バケツを空にする  */
             Buckets[b].rear  :=  0 ;
     }
     while (   (↑ptr_A).front  ≠  0   ) {
             doc_no  :=  CellSpace[(↑ptr_A).front].element ;          /*  キューA の先頭にある書類の番号  */
             if (   b_size = 31   )
                     b  :=  Documents[doc_no].day ;                   /*  書類番号 doc_no  の日  */
              else      b  :=  Documents[doc_no].month ;              /*  書類番号 doc_no  の月  */
        /*  キューA の先頭のセルを Dequeue し, Buckets[b]  に Enqueue する  */
             if (   Buckets[b].front  = 0   )              /*  Buckets[b]  に Enqueue する最初の要素である  */
                     Buckets[b].front   :=   (↑ptr_A).front ;
             else     CellSpace[ Buckets[b].rear ].next   :=   (↑ptr_A).front ;
             Buckets[b].rear   :=   (↑ptr_A).front ;
             (↑ptr_A).front   :=   CellSpace[ (↑ptr_A).front ].next ;
             CellSpace[ Buckets[b].rear ].next   := 0   ;                    /*  Dequeue, Enqueue の終了  */
     }
     for (   1 ≤ b ≤ b_size  なる各 b.  b  の昇順に行う   ) {
             if (   Buckets[b].front  ≠  0   ) {                            /*  Buckets[b]  は要素を持つ  */
                /*  キューA の後ろにキューBuckets[b]  を連接する  */
                     if (   (↑ptr_A).front = 0   )                          /*  キューA  は空である  */
                             (↑ptr_A).front   :=   Buckets[b].front ;
                     else     CellSpace[ (↑ptr_A).rear ].next   :=   Buckets[b].front ;
                     (↑ptr_A).rear  :=   Buckets[b].rear ;                  /*  連接の終了  */
     }       }
} /*   Bucket_Sort   */
```