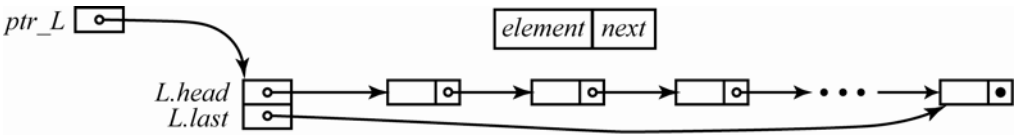
アルゴリズムとデータ構造1　　演習問題 (3)　　回答

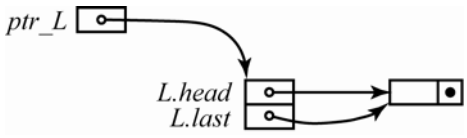
下の関数を呼ぶ方法.

Append(*newdata*, &*L*) ;
DeleteFirst(&*L*) ;

elementtype の要素を *element* 欄に蓄えた連結ﾘｽﾄが次のように定義されている.

```
typedef struct {
            elementtype   element ;      /* 連結ﾘｽﾄに蓄える要素を入れる欄 */
            CELL↑         next ;         /* 次の CELL 型のセルへのポインタ */
      } CELL ;
typedef     CELL↑         position ;     /* position は CELL 型データへのポインタ型 */
typedef struct {
            position      head ;         /* CELL 型データへのポインタ */
            position      last ;         /* CELL 型データへのポインタ */
      } LIST ;

LIST   L ;

/* 連結ﾘｽﾄ L の欄 L.head は, element 欄に意味の無いデータが入っているヘッダセルを指し */
/* 欄 L.last は, element 欄に最後の要素を持つ CELL 型のセルを指す */
/* この最後のセルの next 欄には NULL が入っている */
/* L が要素を持たないとき, L.head および L.last は共にヘッダセルを指す */
```





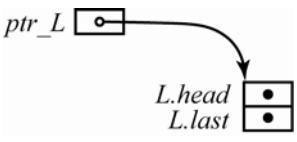

```
void Append( elementtype x,   LIST↑ ptr_L )
    /* 連結ﾘｽﾄ ↑ptr_L の最後の要素として x を付加する */
{
    position      q ;                  /* CELL 型データへのポインタで局所変数 */

    q := new( CELL ) ;                 /* 新たな CELL 型領域を確保し, q がそこを指すようにする */
    (↑(↑ptr_L).last).next := q ;       /* 連結ﾘｽﾄ ↑ptr_L の最後に新たに生成したセルを追加する */
    (↑ptr_L).last := q ;               /* 連結ﾘｽﾄ ↑ptr_L の last が新たに生成したセルを指すようにする */
    (↑q).next := NULL ;                /* 連結ﾘｽﾄ ↑ptr_L の最後のセルの next 欄に NULL を入れる */
    (↑q).element := x ;                /* 最後のセルの element 欄に x を入れる */
} /* Append */
```

```
void DeleteFirst( LIST↑ ptr_L )
    /* 連結リスト ↑ptr_L の最初の要素を取り除く */
{
    position      q ;                          /* CELL 型データへのポインタで局所変数 */

    q := (↑ptr_L).head ;                       /* q はヘッダセルを指す */
    if ( (↑q).next = NULL ) {
        "連結リストは空であると出力する" ;
    }
    else {
        (↑ptr_L).head := (↑q).next ;           /* head がヘッダセルの次のセルを指すようにする */
        free( q ) ;                            /* これまでのヘッダセルを free する */
    }
} /* DeleteFirst */
```

上と同じように, データ型 CELL, position, LIST を定義し, LSIT 型の変数 *L* を定義したとしても, セルの使い方 (意味) を変えれば, プログラムも変えねばならない. 以下では, ヘッダセルを用いない場合の関数を作成する.

```
/* 連結リスト L の欄 L.head および L.last は, それぞれ最初および最後の要素が入っているセルを指す */
/* L.last が指す最後のセルの next 欄には NULL が入っている */
/* L が要素を持たない空リストのとき, L.head および L.last は共に NULL となっている */
/* なお, ここでは使わないが, 空リストのときの Tail(L) は NULL と定めることにしよう */
```

```
void Append( elementtype x,  LIST↑ ptr_L )
    /* 連結リスト ↑ptr_L の最後の要素として x を付加する */
{
    position      q ;                          /* CELL 型データへのポインタで局所変数 */

    q := new( CELL ) ;                         /* 新たな CELL 型領域を確保し, q がそこを指すようにする */
    if ( (↑ptr_L).head = NULL )                /* 連結リスト↑ptr_L は空である */
        (↑ptr_L).head := q ;                   /* 連結リストの先頭に新たに生成したセルを追加する */
    else                                       /* 連結リスト↑ptr_L は要素を持つ */
        (↑(↑ptr_L).last).next := q ;           /* 連結リストの最後に新たに生成したセルを追加する */
    (↑ptr_L).last := q ;                       /* 連結リスト L の last が新たに生成したセルを指すようにする */
    (↑q).next := NULL ;                        /* 連結リスト ↑ptr_L の最後のセルの next 欄に NULL を入れる */
    (↑q).element := x ;                        /* 最後のセルの element 欄に x を入れる */
} /* Append */
```



```
void  DeleteFirst( LIST↑  ptr_L )
    /*  連結リスト ↑ptr_L の最初の要素を取り除く  */
{
    position        q ;                          /*  CELL 型データへのポインタで局所変数  */

    if ( (↑ptr_L).head  =  NULL  )  {
        “連結リストは空であると出力する”;
    }
    else {
        q  :=  (↑ptr_L).head ;                   /*  q  は連結リストの先頭のセルを指す  */
        (↑ptr_L).head  :=  (↑q).next ;           /*  head  がヘッダセルの次のセルを指すようにする  */
        if ( (↑q).next  =  NULL  )               /*  連結リストは要素を 1 個しか持っていない  */
            (↑ptr_L).last  :=  NULL ;            /*  連結リストを空リストの状態にする  */
        free(  q  ) ;                            /*  これまで先頭だったセルを free する  */
    }
} /*  DeleteFirst  */
```