5.4 广义表

- 广义表是线性表的推广。广泛地用于人工智能领域的表处理语言LISP语言。

广义表：是$n \geq 0$个元素的有限序列，记作

$$LS = (\alpha_1, \alpha_2, \cdots, \alpha_n)$$

其中：α_i 或为原子项(原子，一般用小写字母表示)；
或为广义表(子表，一般用大写字母表示)。
n 为广义表的长度。

- 原子：是作为结构上不可分割的成分，它可以是一个数或一个结构。

□ 表头与表尾：LS不为空时，称α_1为表头(head)，称其余元素组成的子表$(\alpha_2, \alpha_3, \cdots, \alpha_n)$为表尾(tail)。

【示例-1】

1）A = ()　　n = 0

2）B = (a, (e))　　n = 2

head(B)= a　　tail(B)=((e))

3）C = ((a,(b,c)), B)　　n = 2

head(C)= (a,(b,c))　　tail(C)= (B)

4）D = (A, a, B)　　n = 3

head(D)= A　　tail(D)= (a,B)

5）E = (a, E) = (a, (a, (a, … ,)))　　n = 2

head(E)= a　　tail(E)= (E)

- 任何一个非空广义表其表头可能是原子或广义表，而其表尾必定为广义表。

结构特点：

1）广义表中的数据元素有相对次序；

2）广义表的长度定义为最外层包含元素个数；

3）广义表的深度定义为所含括弧的重数；

注意：“原子”的深度为 0；“空表”的深度为 1。

4） 广义表可以共享；

5） 广义表可以是一个递归的表。

递归表的深度是无穷值，长度是有限值。

5.5 广义表的存储结构

由于广义表LS=(α_1,α_2,α_3,…α_n)中的数据元素可以具有不同的结构（或是原子，或是广义表），因此，难以用顺序存储结构表示，通常采用链式存储结构来表示。

- ① 头、尾链表存储结构;
- ② 扩展线性链表存储结构。

存储方式

① 头、尾链表存储结构

每个元素用一个结点表示，需要用两种结构的结点：

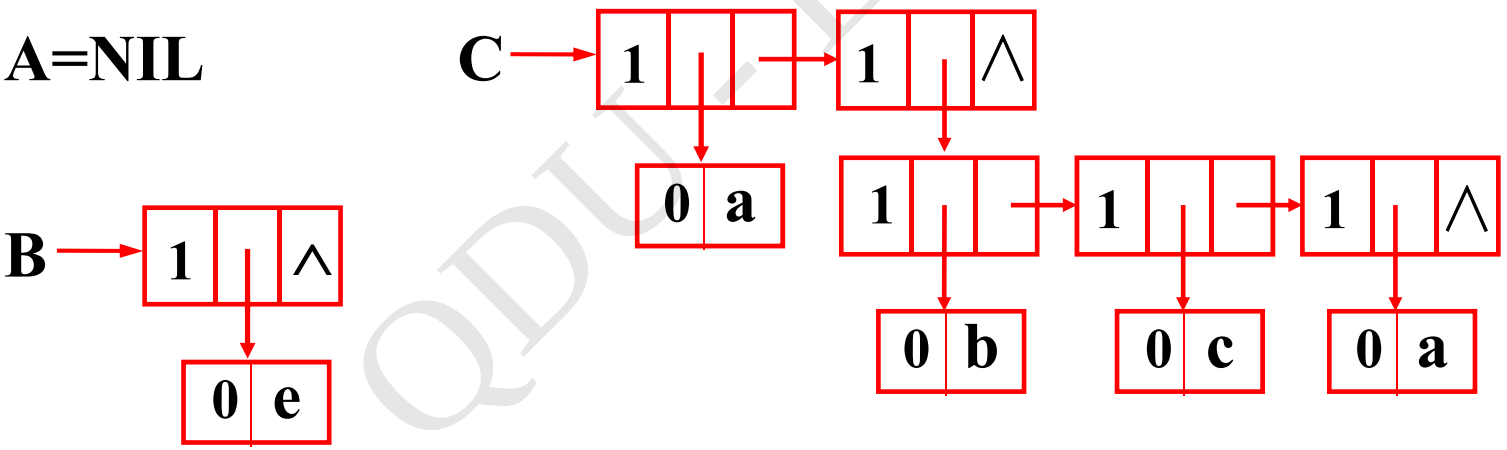
- 表结点
  - 标志域
  - 表头指针
  - 表尾指针
- 原子结点
  - 标志域
  - 值域

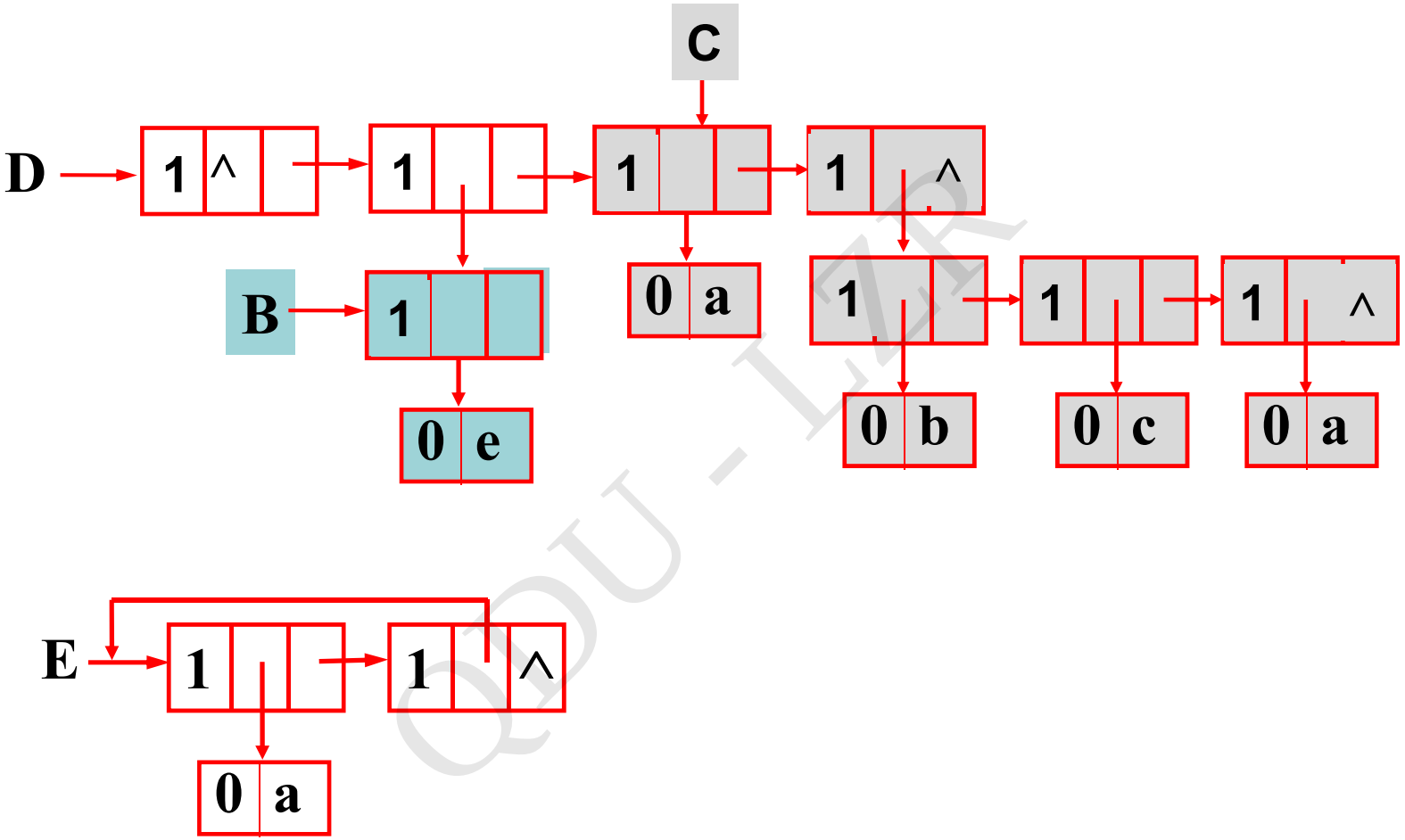
tag=1	hp	tp

tag=0	atom

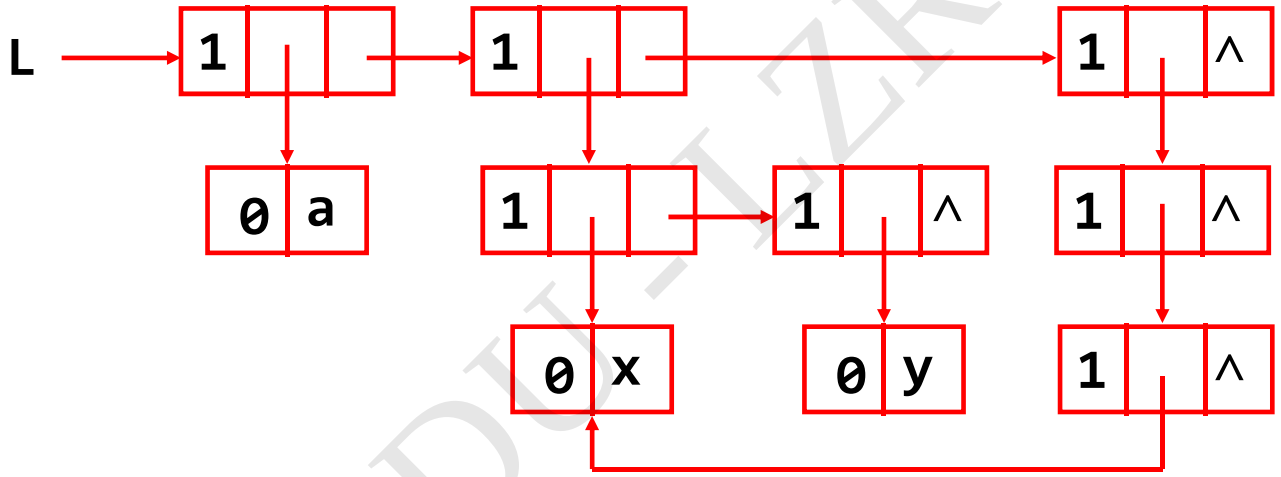
```
// ------广义表的头尾链表存储表示---------
enum ElemTag {ATOM, LIST}; // ATOM==0: 原子, LIST==1: 子表
typedef struct GLNode {
    ElemTag tag;            // 公共部分，用于区分原子结点和表结点
    union {                 // 原子结点和表结点的联合部分
        AtomType atom;      // atom是原子结点的值域，AtomType由用户定义
        struct {
            GLNode *hp, *tp;
        } ptr;      //ptr是表结点的指针域，prt.hp和ptr.tp分别指向表头和表尾
    };
} *GList, GLNode; // 广义表类型
```

【示例-2】已知有广义表 A = ( ), B = (e), C = (a,(b,c,a)), D = (A, B, C), E = (a, E) = (a, (a, (a, … ,))))。试画出其链式存储结构示意图。





【示例-3】 L=(a,(x,y),((x)))



存储方式

② 扩展线性链表存储结构

每个元素用一个结点表示，需要用两种结构的结点：

- 表结点
  - 标志域
  - 表头指针
  - 表尾指针
- 原子结点
  - 标志域
  - 值域
  - 表尾指针

tag=1	hp	tp

tag=0	atom	tp

② 扩展线性链表存储结构

```
// ------ 广义表的扩展线性链表存储表示 -------
enum ElemTag {ATOM, LIST}; // ATOM==0: 原子, LIST==1: 子表
typedef struct GLNode1 {
    ElemTag tag;              //公共部分，用于区分原子结点和表结点
    union {                   // 原子结点和表结点的联合部分
        AtomType atom;        // 原子结点的值域
        GLNode1 *hp;          // 表结点的表头指针
    };
    GLNode1 *tp;              //相当于线性链表的next，指向下一个元素结点
}*GList1, GLNode1;            //广义表类型GList1是一种扩展的线性链表
```

【示例-4】回答下面的问题：

(1) 请将香蕉(banana)用工具 Head()、Tail()从 L 中取出。

L=(apple, (orange, (strawberry, (banana)), peach), pear)

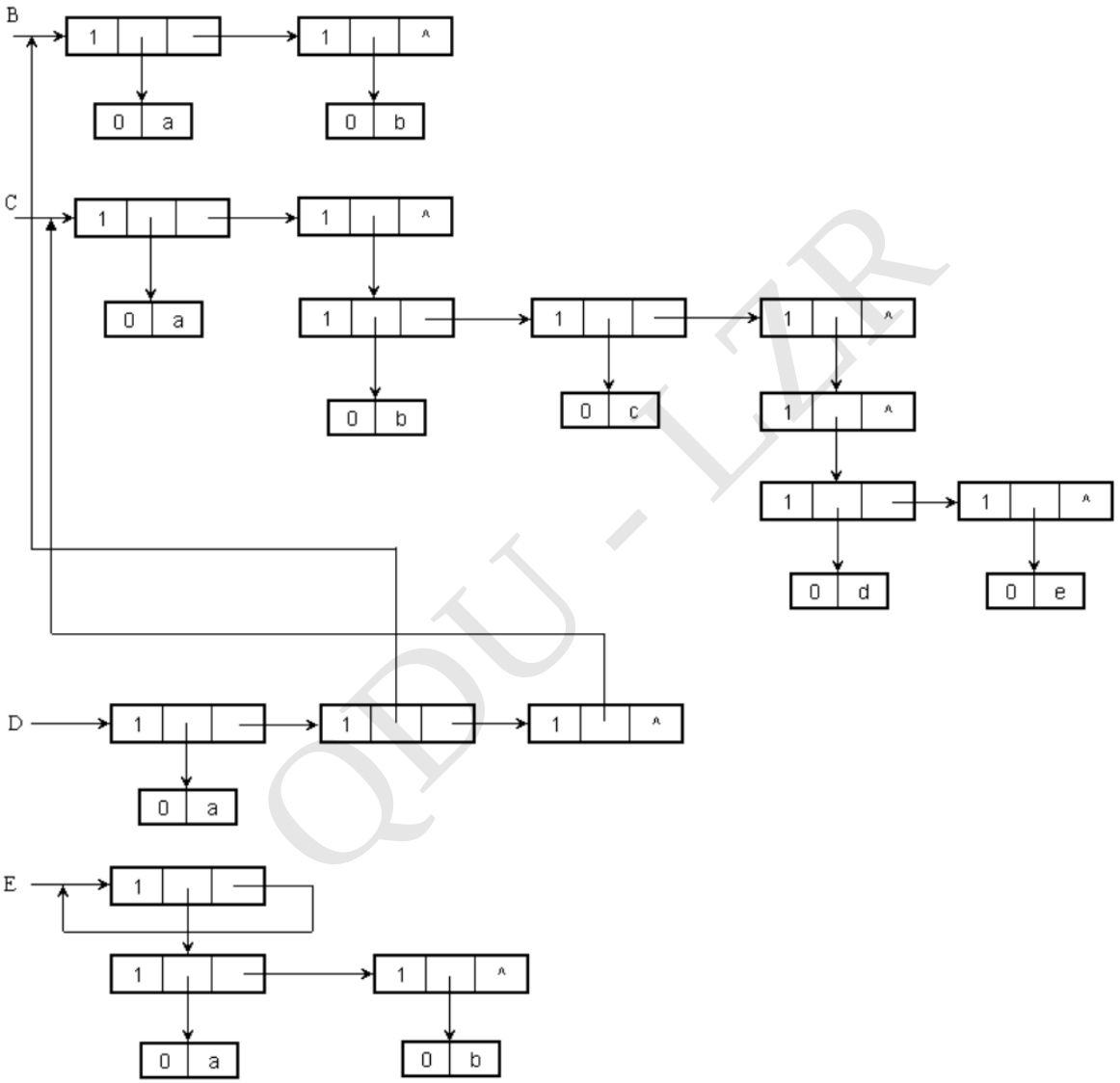
(2) 写出广义表 B=(a, b),C=(a, (b, c, (d,e))), D=(a, B, C), E=((a, b), E)的存储结构(任一种存储方法均可)

解答：

(1) 函数表达式为：

```
head(head(tail(head(tail(head(tail(L)))))))
```

(2) 存储结构为：



END