# 2000年华东计算技术研究所数据结构与程序设计试题



2000年华东计算技术研究所（ECI）数据结构与程序设计试题

# 华东计算技术研究所（ECI）

## 2000年攻读硕士学位研究生入学考试试卷

## 考试科目：数据结构与程序设计

## 一、请完善下列程序，每小题在Pascal语言（a）、C语言（b）中任选一题。若都做，按（a）计分。（共40分）

注意：每个空格填一个表达式或一个语句。

1. 下列算法为奇偶交换排序，思路如下：第一趟对所有奇数的i，将a[i]和a[i+1]进行比较，第二趟对所有偶数的i，将a[i]和a[i+1]进行比较，每次比较时若a[i]>a[i+1]，将二者交换；以后重复上述二趟过程，直至整个数组有序。

程序(a)

```
Procedure oesort(var a : array[1..n] of integer),
Var  flag : boolean;          i ,t : integer;
Begin
    Repeat
        flag := false;
        for i := 1 to n step 2 do
            if (a[i]>a[i+1]) then
            [   flag := _____ (1) _____,
                t := a[i+1];     a[i+1] := a[i];
                _____ (2) _____,
            ]
        for i := _____ (3) _____ do
            if (a[i]>a[i+1]) then
            [   flag := _____ (4) _____,
                t := a[i+1];     a[i+1] := a[i];
                a[i] := t;
            ]
    Until _____ (5) _____,
End;
```

程序（b）

```
Void oesort( int a[n])
{    int flag, i. t;
     do {
        flag = 0;
        for (i = 1;i<n;i++,i++)
           if (a[i]>a[i+1])
           {  flag = _____ (1) _____,
              t = a[i+1];     a[i+1] = a[i],
              _____ (2) _____,
           }
        for _____ (3) _____
           if (a[i]>a[i+1])
           {  flag = _____ (4) _____,
              t = a[i+1];     a[i+1] = a[i],
              a[i] = t;
           }
     } while (_____ (5) _____),
}
```



2. 下列算法实现求采用顺序结构存储的串 s 和串 t 的一个最长公共子串。

程序（a）

```
Procedure maxcomstr(var s,t : orderstring; var index, length : integer);
Var i,j,k,length1 : integer;       con : boolean;
Begin
   index := 0;    length := 0;  i := 1;
   while (i<=s.len) do 【
      j := 1;
      while (j<=t. len) do【
         if (s[i]=t[j]) then 【
            k := 1;       length1 := 1; con := true;
            while con do
               if _____ (1) _____ then 【
                  lenght1 := length1+1; k:= k+1;
               】
               else  _____ (2) _____,
```

```
            if (length1>length) then 【
               index := i;       length := length1;
            】
            _________(3)_________;
         】
         else _________(4)_________;
      】
      _________(5)_________;
   】
End;
```

程序 (b)

```
Void maxcomstr(orderstring *s, *t; int index, length)
{   int i,j,k,length1,con;
   index = 0;      length = 0;       i = 1;
   while (i<=s.len) {
      j = 1;
      while (j<=t.len) {
      if (s[i]==t[j]) {
            k = 1;    length1 = 1;      con = 1;
            while (con)
               if _______(1)_______{
                  lenght1 = length1+1;     k= k+1;
               }
               else   _______(2)_______;
            if (length1>length) {
               index = i,    length = length1;
            }
            _________(3)_____________;
         }
         else _____(4)_______;
      }
      _________(5)_________;
   }
```

3 设 t 是给定的一棵二叉树，下面的递归程序 count(t)用于求得：二叉树 t 中具有非空的左、右两个儿子的结点个数 N2；只有非空左儿子的结点个数 NL；只有非空右儿子的结点个数 NR 和叶子结点个数 N0。N2、NL、NR、N0 都是全局量，且在调用 count(t)之前都置为 0。

程序（a）

```
type nodeptr = ^nodetype,
nodetype = record
                data : integer;
                lchild, rchild : nodeptr;
           end;
var  N2, NL, NR, N0 : integer;
Procedure count(t : nodeptr);
Begin
    if t^.lchild<>nil then
        if _____(1)_____ then
            N2 := N2+1
        else
            NL := NL+1
    else
        if _____(2)_____ then
            NR := NR+1
        else
            _____(3)_____;
    if t^.lchild<>nil then
        _____(4)_____;
    if t^.rchild<>nil then
        _____(5)_____;
End;
(* call form : if t<>nil then count(t); *)
```

程序（b）

```
typedef struct node{
                    int data;
                    struct node *lchild, *rchild;
                  } NODE;
int   N2, NL, NR, N0;
```

```
void count(NODE *t)
{
    if (t->lchild != NULL)
        if _____(1)_____
            N2 ++;
        else
            NL ++;
    else
        if _____(2)_____
            NR ++;
        else
            _____(3)_____;
    if (t->lchild != NULL)
        _____(4)_____;
    if (t->rchild != NULL)
        _____(5)_____;
}
/* call form : if (t !=NULL) count(t); */
```

4. 下面是一个求两个集合 A 和 B 之差 C=A-B 的程序，即当且仅当 e 是 A 中的一个元素，但不是 B 中的一个元素时，e 才是 C 中的一个元素。集合用有序链表实现，初始时，A、B 集合中的元素按递增排列，C 为空；操作完成后，A、B 保持不变，C 中元素按递增排列。下面的函数 append(last, e)是把值为 e 的新结点链接在由指针 last 指向的结点的后面，并返回新结点的地址；函数 difference(A, B)实现集合运算 A-B，并返回表示结果集合 C 的链表的首结点的地址。在执行 A-B 运算之前，用于表示结果集合的链表首先增加一个附加的表头结点，以便新结点的添加，当 A-B 运算执行完毕，再删除并释放表示结果集合的链表的表头结点。

程序（a）

```
type nodeptr = ^nodetype;
nodetype = record
                element : integer;
                link : nodeptr;
            end;
var   A, B, C : nodeptr;
```

```
Function append(last : nodeptr; e : integer) . nodeptr;
Begin
    new(last^.link);          last^.link^.element := e;
    append := last^.link
End;

Function difference(A, B : nodeptr) : nodeptr;
Var   C, last : nodeptr;
Begin
    new(C);        last := C;
    while ___________(1)___________ do
        if A^.element < B^.element then
        【     last := append(last, A^.element);
               A := A^.link;
        】
        else
               if _________(2)_____________ then
               【    A := A^.link;          B := B^.link;  】
               else
                    ________(3)_____________;
    while ___________(4)___________ do
        【     last := append(last, A^.element);
               A := A^.link;
        】
    ___________(5)___________;
    last := C;            C := C^.link;
    dispose(last);        difference := C
End;
(* call form : C := difference(A, B); *)
```

程序（5）

```
typedef struct node{
                        int element;
                        struct node *link;
                      } NODE;
NODE *A, *B, *C,
```

```
NODE *append(NODE *last, int e)
{
    last->link = (NODE *)malloc(sizeof(NODE));
    last->link->element = e;
    return(last->link);
}

NODE *difference(NODE *A, NODE *B)
{
    NODE *C, *last;

    C = last = (NODE*)malloc(sizeof(NODE));
    while ___________(1)___________
        if (A->element < B->element)
        {
            last = append(last, A->element);
            A = A->link;
        }
        else
            if _________(2)___________
            {
                A = A->link;       B = B->link;
            }
            else
                ________(3)_____________;
    while ___________(4)___________
        {
            last = append(last, A->element);
            A = A->link;
        }
    ___________(5)___________;
    last = C;              C = C->link;
    free(last);            return(C);
}
/* call form : C = difference(A, B); */
```

## 二、请完善下列算法。（共 24 分）

注意：每个空格填一个表达式或一个语句

1．已知一个循环单链表如图 1 所示，下列算法（图 2 所示）实现将该链表所有箭头方向取反。

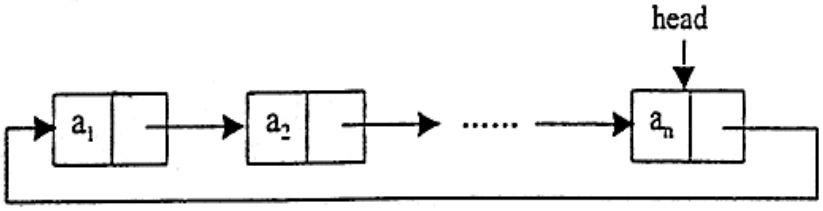


图 1　循环单链表结构图

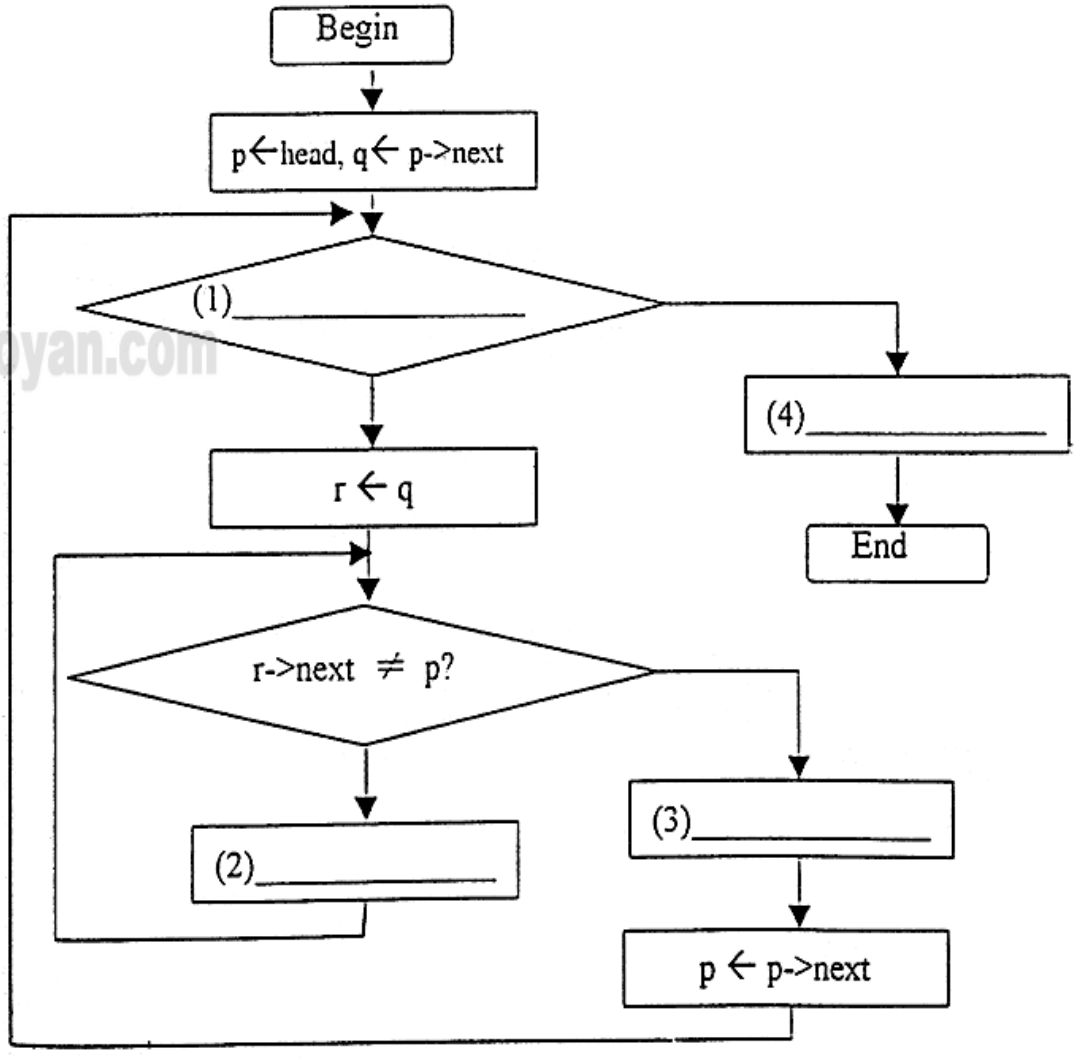


图 2 循环单链表逆置算法

2. 设 t 是一棵非空的查找树，a 和 b 是树 t 中某两个结点的值。下面的非递归算法（如图 3 所示）用于判别值为 a 和 b 的两个结点是否同在查找树 t 中的一根树枝上。若同在一根树枝上，则返回 1（true）；否则返回 0（false）。所谓两个结点同在一根树枝上是指两个结点同在树中一条自上而下的路径上。

注：查找树采用二叉链表表示法，与第一题中第 3 题的表示法相同。

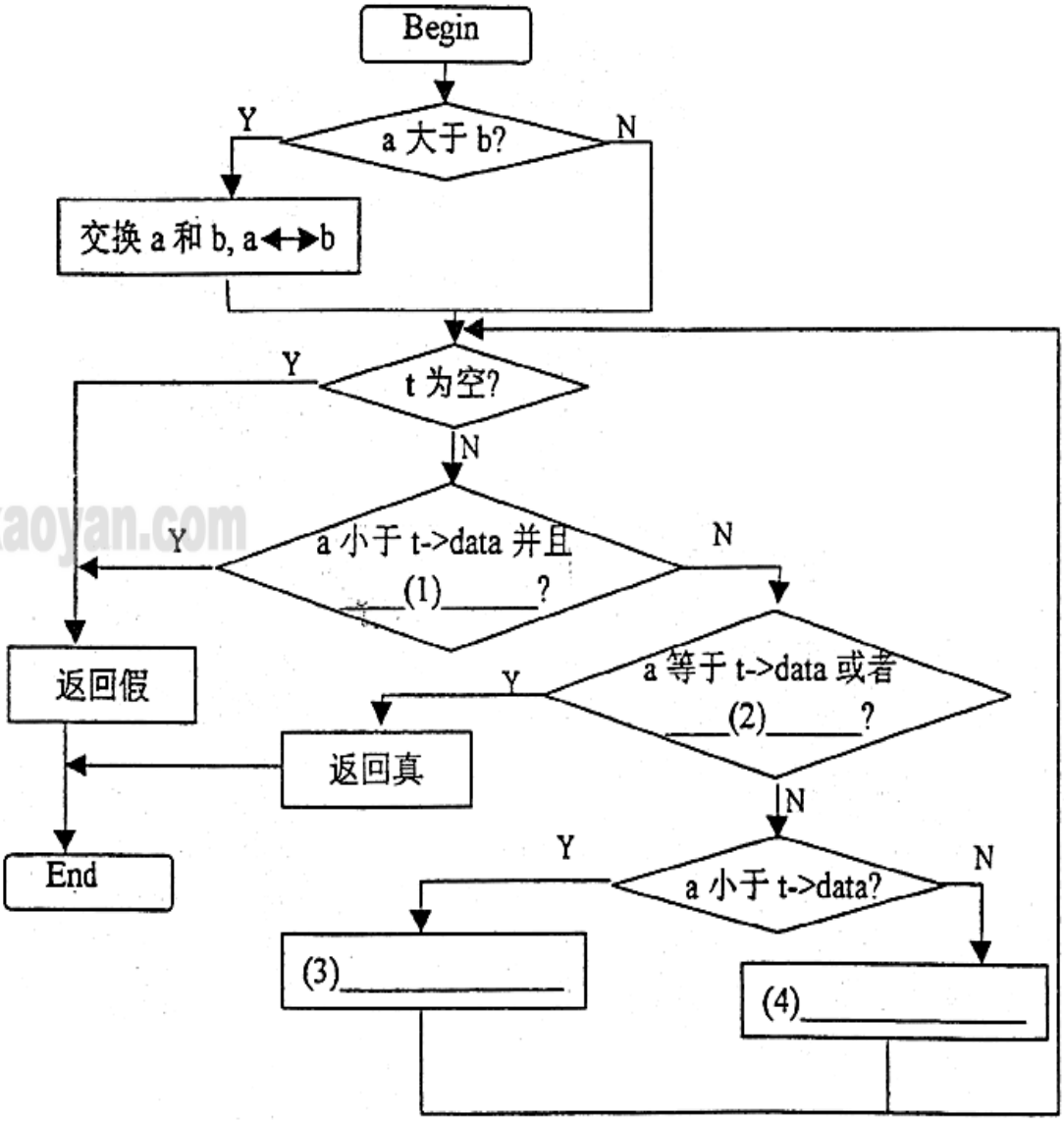


图 3 判别两个结点是否同在查找树的一根树枝上的算法

## 三、 请编写完整的程序。（20分）

如果矩阵 A 中存在这样的一个元素 A[i,j]满足条件：A[i,j]是第 i 行中值最小的元素，且又是第 j 列中值最大的元素，则称之为该矩阵的一个马鞍点。请编程计算出 m*n 的矩阵 A 的所有马鞍点。

## 四、 请用流程图或类高级语言（pascal 或 c）表示算法。（16分）

已知有向图有 n 个顶点，请写算法，根据用户输入的偶对建立该有向图的邻接表。即接受用户输入的$\langle v_i, v_j\rangle$（以其中之一为 0 标志结束），对于每条这样的边，申请一个结点，并插入到 $v_i$ 的单链表中，如此反复，直到将图中所有边处理完毕。

提示：先产生邻接表的 n 个头结点（其结点数值域从 1 到 n）。