2002 攻读硕士学位研究生入学考试试题

报考专业：**计算机系统结构 计算机软件与理论 计算机应用技术**
报考方向：**以上专业各方向**
考试科目：**高级语言程序设计**
注意：
1 答案一律书写在答题纸上
2 题签随答题纸交回
3 可选用 C 或 PASCAL 语言之一编写程序
【一】（20 分）
设〈N[size=-1]1,N[size=-1]2,……，N[size=-1]k〉是一个整数序列，若满足条件
N[size=-1]1≦N[size=-1]2≦……≦N[size=-1]k-1,1≦k
则称上述序列为准递增序列。这时若把 Nk 插入到<
N[size=-1]1,N[size=-1]2,……,N[size=-1]k-1>中，使得插入后所得的序列<
N[size=-1]1',N[size=-1]2',……,N[size=-1]k-1'>仍然保持递增性，则我们称这种插入为保序插入。试写出一个函数（或过程）insert,使得它只要给出一个准递增序列和其长度，则将对其进行包叙插入，并返回保序插入后的新序列。在写程序时，要求把 insert 写成递归的形式。
[参考答案]

```
/*==========================================================
==================================*/
/*函数名称：2002_1.c
*/
/*程序目的：递归地将一个数插入一个准递增序列，且插入完毕后仍然保持准递增特性
*/
/*Writen by Apechn ,Soft Lab of JLU
*/
/*==========================================================
==================================*/
#include <stdio.h>
struct node //定义链表结点结构，在最前面给出
{
  int num;
  node* next;
}
node* insert(node* head,int n)//题目要求的函数
{
  node *p;
  if (n <= head->num) //若 n 已经到了准递增序列的合适的位置，递归出口
  {
    p = (node*)malloc(sizeof(node));//新建一个节点用来存放 n
    p->num = n;
    p->next = head;
    head = p;
```

```
  }
  else
    insert(head->next,n);//否则递归插入
  return head;
}
```

【二】（20 分）

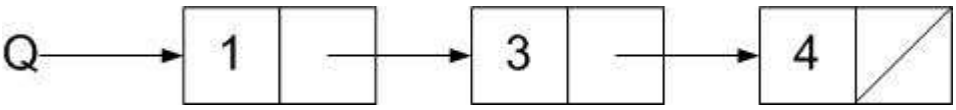
假设有整数序列〈N[size=-1]1,N[size=-1]2,……，N[size=-1]k〉，则我们称其中的子序列〈N[size=-1]i,N[size=-1]i+1,……，N[size=-1]j〉为上述序列的递增子序列，如果有 Nk≦Nk+1,其中 i≦k≦j。试写一个函数 MaxLeng，使得它对任给的整数序列，返回一个整数，它表示给定序列的最长递增子序列的长度。例如，假设有<7,2,3,4,2,2,5>,则其最长递增子序列的长度是 3。在函数 MaxLeng 中，要求序列中的每个元素不能被扫描一次以上。

[参考答案]

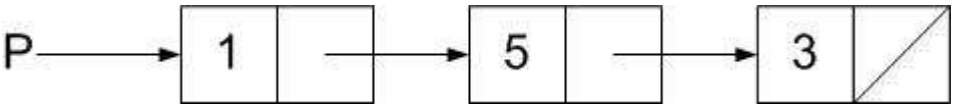
```
/*==========================================================
==========================*/
/*函数名称：2002_2.c
*/
/*函数目的：求给定序列的最长递增子序列的长度
*/
/*Writen by Apechn,Soft Lab of JLU                                    */
/*==========================================================
==========================*/

#include <stdio.h>
const int N = 8;
int MaxLeng(int a[])
{
    int i,len = 1,lmax = 1;
    for (i = 1;i < N;i++)
    {
      if(a >= a[i-1]) //如果满足递增子序列的定义，len 增一
      {
        len ++;
        contnue;
      }
      if (lamx < len)
        lmax = len;
      len = 1;
    }
    return lmax;
}
```

【三】（20 分）

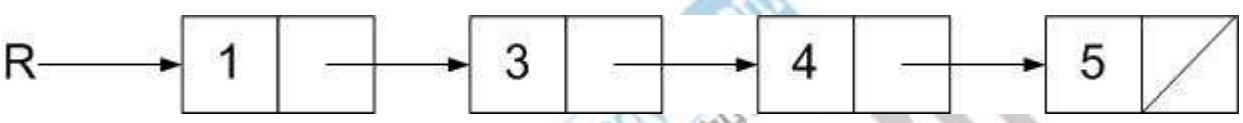
假设用链表表示集合，例如集合{1，3，4}可表示为下面链表：



要求写一个函数 SetAdd，它有两个参数 P 和 Q，他们分别指向两个链表（表示集合，每个没有相同元素），执行函数调用 SetAdd(P,Q)后将返回链表 R，R 是表示 P 集合加 Q 集合所得集合的链表（集合加即集合并）。例如，再有 P 链表为：



则执行 SetAdd(P,Q),结果应返回下面链表：



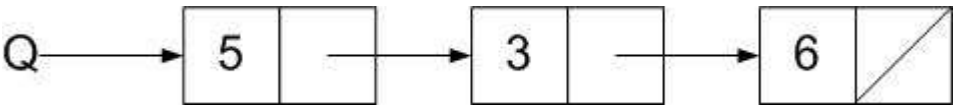
[参考答案]

```
/*==========================================================
==========================*/
/*函数名称：2004_3.c
*/
/*函数目的：用链表表示集合的相加
*/
/*Writen by Apechn ,Soft Lab of JLU                                    */
/*==========================================================
==========================*/

#include <stdio.h>
struct node //定义链表结点结构，在最前面给出
{
   int num;
   node* next;
}
int ismember(int a,node* head)//子函数，判断整数 a 是不是在以 head 为头结点的链表中
{
   int sign = 0; //如果在，返回 1，否则返回 0
   node* p = head;

   if (p == NULL) //注意集合为空集的情况
      return sign;

   while (p != NULL)
   {
      if (p->num == a)
      {
         sign = 1;
```

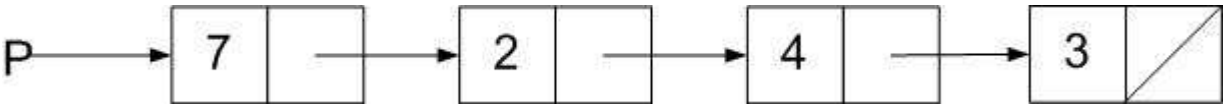
```
            break;
        }
        p = p->next;
    }

    return sign;
}
node* SetAdd(node* P,node* Q)//题目要求的函数
{
    int n;
    node *r,*tail,*point1,*poing2;
    if (P == NULL)
        return Q;

    if (Q == NULL)
        return P;
    r = P;
    while (r != NULL) //tail 用来记录 P 链表的尾部
    {
        tail = r;
        r = r->next;
    }
    r = Q;
    while (ismember(r->num,P))//找到 Q 中第一个不在 P 中的元素的位置，记为 r
        r = r->next;
    point1 = r;
    poing2 = r->next;
    while (point2 != NULL) //从 r 开始遍历链表 Q
    {
        n = point2->num;
        if(ismember(n,P))
            point1->next = point2->next;//如果遍历到的元素是 P 中的元素，那么把它删掉
        else
            point1 = point2;//否则继续
        point2 = point2->next;
    }
    tail->next = r; //把剩余的元素放到 P 中
    return P;
}
```

【四】（20 分）

假设用链表表示八进制数，如八进制数 536 被表示为下面链表：



要求写一个函数 Add，它有两个参数 P 和 Q，分别指向表示八进制数的链表。执行函数调用 Add(P,Q)后，将返回表示 P 八进制数加 Q 八进制数所得数的链表 R。例如，假设再有 P 链表:



则执行 Add(P,Q),结果应返回下面链表:

[参考答案]

```
/*==========================================================
==========================*/
/*函数名称：2004_4.c
*/
/*函数目的：用链表表示八进制数以及它们的和
*/
/*Writen by Apechn ,Soft Lab of JLU                                    */
/*==========================================================
==========================*/
#include <stdio.h>
struct node
{
   int num;
   node* next;
};
node* inverse(node* head) //子函数，逆转链表的所有指针，把原来的尾结点变成头结点
{
   node *f,*g,*h;
   if (head != NULL)
   {
      f = head;
      g = NULL;
      while (f->next != NULL)
      {
         h = f;
         f = f->next;
         h->next = g;
         g = h;
```

```
    }
    f->next = g;
    head = f;
  }
  return head;
}
node* Add(node* P,node* Q) //题目要求的函数
{
  node *p,*q,*r,*head = NULL;
  int carry,temp;
  if (P == NULL)
    return Q;
  if (Q == NULL)
    return P;
  p = inverse(P); //逆转两个链表的指针
  q = inverse(Q);
  carry = 0; //进位设为 0
  while (p != NULL && q != NULL)//逐为相加两个数，知道其中一个加完为止
  {
    temp = p->num + q->num +carry;
    r = new node;
    r->num = temp % 8;
    r->next = head;
    head = r;
    carry = temp/8;
    p = p->next;
    q = q->next;
  }
  if (q == NULL) //修改一下指针，便于统一计算
  q = p;
  while (q != NULL) //把剩余的数加完
  {
    temp = q->num + carry;
    r = new node;
    r->num = temp % 8;
    r->next = head;
    head = r;
    carry = temp/8;
    q = q->next;
  }
  return head;
}
```

【五】（20 分）

假设有某种语言的函数定义

```
Function f(x:real；y:real):real
        begin L:S1；
                S2；
                x ← x + 1.5；
                y ← y + x；
                if y≤100 then goto L
                    return(sin(y))
        end
```

其中←表示赋值操作，S1 和 S2 表示语句，其中没对 x 和 y 的赋值，也不含 goto 语句。要求把函数 f 的定义改写成递归函数的形式。函数 f 有两个实型形参，计算结果是返回一个实数，即返回 sin(y)的值。

[参考答案]

```
/*==========================================================
==================================*/
/*函数名称：2002_5.c
*/
/*程序目的：将一个函数改写为递归函数
*/
/*Writen by Apechn ,Soft Lab of JLU
*/
/*==========================================================
==================================*/

#include <stdio.h>
#include <math.h>
int m,n;
doublef1(double x,double y)
{
  L:
  printf("This is S1 %d times \n",m);
  m++;
  printf("This is S2 %d times \n",n);
  n++;
  x = x + 1.5;
  y = y + x;
  if (y <= 100)
  goto L;
  return sin(y);
}
double f3(double x,double y)//递归函数
{
  printf("This is S1 %d times\n",m);
```

```
    m++;
    printf("This is S2 %d times\n",n);
    n++;
    x = x + 1.5;
    y = y + x;
    if (y <= 100) //递归出口
        return f3(x,y);
    else
        return sin(y);
}
```