2005 年计算机学院研究生复试试题

（机试部分）

说明：试题 1，2 必做，试题 3，4 中任选一个，考试时间 14:30-18:00。

1　归并两个有序的顺序表，要求两个有序顺序表的数据从文件读入，归并后的数据输出到结果文件中。（30 分）

顺序表 A 的数据为：2，6，8，9，14，20，22

顺序表 B 的数据为：3，6，9，15

```
#include <stdio.h>
#define maxLen 100
typedef struct _SeqList
{
    int Data[maxLen];
    int size;
};
typedef struct _SeqList SeqList;
typedef struct _SeqList* pSeqList;
LoadData(char *fname, pSeqList la)
{
    FILE *fp;
    int temp;
    int i=0;
    fp = fopen(fname, "rt");
    while (!feof(fp))
    {
        fscanf(fp, "%d\n", &temp);
        la->Data[i++] = temp;
    }
    la->size = i;
    fclose(fp);
}
OutputData(char *fname, SeqList la)
{
    FILE *fp;
    int i;
    fp = fopen(fname, "wt");
    for (i=0; i<la.size; i++)
        fprintf(fp, "%d\n", la.Data[i]);
    fclose(fp);
}
/*
    Implement the following parts as ToDo
```

```
*/
MergeList( SeqList la, SeqList lb, pSeqList lc)
{
        /*    ToDo*/
}
main()
{
    SeqList la, lb, lc;
    LoadData("a.txt", &la);
    LoadData("b.txt", &lb);
    MergeList(la, lb, &lc);
    OutputData("c.txt", lc);

}
```

2　阅读下列函数说明和 C 代码，将应填入其中 _____处的字句，写在答案的对应栏内。随后上机调试程序，得到相应结果。（30 分）

读入一批以负数为结束的正整数，数据为 5，7，2，4，9，－1，建立一个带头结点的链表，链表的每个结点中包含有两个指针：一个用于链接输入的先后顺序，另一个用于链接输入整数从小到大的顺序。并分别按两个指针方向进行遍历输出。

```
#include <stdio.h>
#include<stdlib.h>
typedef struct NODE
{    int val   ;
     struct NODE *order;
     struct NODE *sort;
}NODE;
void main( )
{    NODE *h,*u,*v,*p,*tail;
     int d;
     h=(NODE *)malloc(sizeof(NODE));
     tail=h;
     tail->sort=NULL;
     printf("Please input data:");
     do
     {          scanf("%d",&d);
                if(d>0)
                {    p=(NODE *)malloc(sizeof(NODE));
                     p->val=d;
                     ____（1）____;
                     tail=p;
                     for(u=h,v=u->sort;v&&v->val<d;____（2）____);
                     p->sort=v;
                     ____（3）____;
                }
     }
     while(d>0);
     ____（4）____;;
     p=h->sort;
     while(p)
     {    printf("%d,",p->val);
          p=p->sort;
     }
     printf("\n\n");
     for (p=h->order;p;p=p->order)
          printf("%d,",p->val);
     printf("\n\n");
}
```

3 已给定先序构造一棵二叉树的算法，请你完成其余部分，包括先序、中序、后序遍历二叉树并打印出来。先序序列建立二叉树的顺序读入字符为 A B C Φ Φ D E Φ G Φ Φ F Φ Φ Φ，注意输入时Φ用空格代替。（40 分）

```
#include <stdio.h>
typedef struct BiTNode
{
    char e;
    struct BiTNode *lchild,*rchild;
}BiTNode;

/*
 Implement the following parts as ToDo
*/

void preOrderTraverse(BiTNode *T1)
{
    /*  ToDo*/
}

void inOrderTraverse(BiTNode *T1)
{
    /*  ToDo*/
}

void postOrderTraverse(BiTNode *T1)
{
    /*  ToDo*/
}

int CreateBiTree(BiTNode **T1)
{

    char ch, a;
    scanf("%c%c", &ch, &a);

    if (ch == '  ')
    {
     /*  ToDo*/
    }
    else
     {
        /*  ToDo*/

        CreateBiTree(&((*T1)->lchild));
```

```
            CreateBiTree(&((*T1)->rchild));
        }
       return 1;
}

main()
{
       /*
       Call CreateBiTree() function
       */
       CreateBiTree();

       /*
       Call three different Traverse functions
       */
       preOrderTraverse();
       inOrderTraverse();
       postOrderTraverse();

}
```

4　根据二叉树的先序和中序序列，设计算法重构出这棵二叉树。（40 分）

已知一棵二叉树的先序遍历序列是 ABECDFGHIJ，中序遍历序列是 EBCDAFHIGJ，请设计一个算法，将这棵二叉树的结构重构出来，并且输出它的三种深度优先遍历的结果（即先序、中序和后序序列）。

```
#include <stdio.h>

typedef struct BiTNode
{
    char e;
    struct BiTNode *lchild,*rchild;
}BiTNode;

/*
 Implement the following parts as ToDo
*/

void preOrderTravse(BiTNode *T1)
{
    /*  ToDo*/
}

void inOrderTravse(BiTNode *T1)
{
    /*  ToDo*/
}

void postOrderTravse(BiTNode *T1)
{
    /*  ToDo*/
}

int CreateBiTree(BiTNode **T1, char *preString, char *inString, int start, int end)
{

    if (start == end)
    {
     /*  ToDo*/
    }
    else
     {
        /*  ToDo*/

        CreateBiTree(&((*T1)->lchild), preString, inString, start, middle);
        CreateBiTree(&((*T1)->rchild) , preString, inString, middle+1, end);
```

```
        }
    return 1;
}

main()
{
    /*
    Call CreateBiTree() function
    */
    CreateBiTree();

    /*
    Call three different Travse functions
    */
    preOrderTravse();
    inOrderTravse();
    postOrderTravse();

}
```