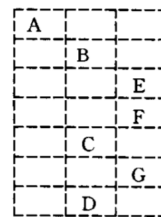
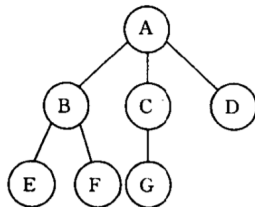


Homework 11.9

和泳毅 PB19010450

```
1 //孩子-兄弟链表
2 typedef struct CSNode{
3     char data; //数据域
4     struct CSNode *fristchild, *nextsibling; //孩子、兄弟
5 }CSNode, *CSTree;
```

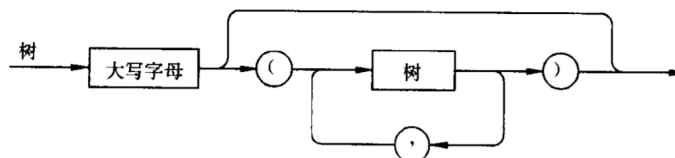
6.71 假设树上每个结点所含的数据元素为一个字母，并且以孩子—兄弟链表为树的存储结构，试写一个按凹入表方式打印一棵树的算法。例如:左下所示树印为右下形状。



类C描述:

```
1 Status PrintCSTree_1(CSTree T, int i){
2     //以孩子-兄弟链表按凹入表打印树, i初始为0用以控制缩进量, 每一层次缩进相同
3     if(T){ //树存在
4         for(n = 1; n <= 2 * i; n++) printf(" "); //一缩进两空格
5         printf("%c\n", T->data);
6         PrintCSTree_1(T->fristchild, i + 1);
7         PrintCSTree_1(T->nextsibling, i);
8     } //if
9     return OK;
10 } //PrintCSTree_1
```

6.74 试写一递归算法,以6.73题给定的树的广义表表示法的字符序列形式输出以孩子—兄弟链表表示的树。



6.71 题中的树可用下列形式的广义表表示:

$A(B(E, F), C(G), D)$

类C描述:

```
1 Status PrintCSTree_2(CSTree T){
2     //以孩子-兄弟链表按广义表格式打印树的递归算法
```

```

3     if(T){
4         printf("%c",T->data);
5         if(T->firstchild){
6             printf("(");
7             for(p = T->firstchild;p;p = p->nextsibling){
8                 PrintCSTree_2(p);
9                 if(p->nextsibling) printf(",");
10            }//for
11            printf(")");
12        }//if_2
13    }//if_1
14    return OK;
15 }//PrintCSTree_2

```

完整C实现:

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <malloc.h>
4  #define OVERFLOW    -1
5  #define OK    1
6  #define Status int
7
8  typedef struct CSNode{
9      char        data;
10     struct CSNode  *firstchild,*nextsibling;
11 }CSNode,*CSTree;
12
13 Status CreateCSTree(CSTree &T){
14     char n;
15     scanf("%c",&n);
16     getchar();
17     if(n == '#') T = NULL;
18     else{
19         T = (CSTree)malloc(sizeof(CSNode));
20         if (!T){
21             printf("建立二叉树时出错。 \n");
22             exit(OVERFLOW);
23         }
24         T->data = n;
25         printf("输入%c的孩子节点: ",n);
26         CreateCSTree(T->firstchild);
27         printf("输入%c的兄弟节点: ",n);
28         CreateCSTree(T->nextsibling);
29     }
30     return OK;
31 }
32
33 Status PrintCSTree_1(CSTree T,int i)
34 Status PrintCSTree_2(CSTree T)
35
36 int main(){
37     CSTree T;
38     printf("输入树的根节点: ");
39     CreateCSTree(T);
40     printf("\n凹入表方式打印\n");

```

```

41     PrintCSTree_1(T,0);
42     printf("\n广义表方式打印\n");
43     PrintCSTree_2(T);
44     return 0;
45 }

```

结果测试:

```

输入树的根节点: A
输入A的孩子节点: B
输入B的孩子节点: E 凹入表方式打印
输入E的孩子节点: # A
输入E的兄弟节点: F    B
输入F的孩子节点: #    E
输入F的兄弟节点: #    F
输入B的兄弟节点: C    C
输入C的孩子节点: G    G
输入G的孩子节点: #    D
输入G的兄弟节点: #
输入C的兄弟节点: D 广义表方式打印
输入D的孩子节点: # A(B(E, F), C(G), D)
输入D的兄弟节点: #
输入A的兄弟节点: #

```