1、输入结点序列为5,10,35,15,10,70,25,30,70，画出在空二叉树基础上逐步插入结点后形成的二叉查找树。

****

2、完成二叉查找树下插入算法的非递归版本。

template <class Record>

Error\_code Search\_tree<Record> :: search\_and\_insertn(

Binary\_node<Record> \* &sub\_root, const Record &new\_data)

{

Binary\_node<Record> \*p=sub\_root;

Binary\_node<Record> \*parent=NULL;

while (p!=NULL){

if (p->data==new\_data)

return duplicate\_error;

else

if (p->data<new\_data){

parent=p;

p=p->right;

}

else{

parent=p;

p=p->left;

}

}

Binary\_node<Record> \*new\_node = new Binary\_node<Record>(new\_data);

if (sub\_root==NULL)

sub\_root=new\_node;

else{

if (parent->data<new\_data)

parent->right=new\_node;

else

parent->left=new\_node;

}

return success;

}

3、设计算法，生成一棵二叉树的镜像二叉树。如下图中左右两棵二叉树互为镜像。

template <class Entry>

Binary\_node<Entry> \*Binary\_tree<Entry>::recursive\_mirror( Binary\_node<Entry> \*sub\_root)

{

if (sub\_root == NULL) return NULL;

Binary\_node<Entry> \*temp = new Binary\_node<Entry>(sub\_root->data);

temp->right = recursive\_mirror(sub\_root->left);

temp->left = recursive\_mirror(sub\_root->right);

return temp;

}

4、假设一个仅包含二元运算符的算术表达式存储在二叉树的二叉链表中，设计算术表达式求值的算法。

int postandcalc(Binary\_node<char> \*sub\_root)

{

if (isdigit(sub\_root->data))

return sub\_root->data-48;

else

{

int x=postandcalc(sub\_root->left);

int y=postandcalc(sub\_root->right);

char op=sub\_root->data;

switch (op){

case '+':return x+y;

case '-':return x-y;

case '\*':return x\*y;

case '/':return x/y;

}

}

}