实 验 报 告

课程名称：数据结构 (C)

实验项目：二叉树的生成与操作

实验仪器：

项目	报告格式	写作质量	逻辑、注释质量、思想描述	复杂度分析	合计
百分比 (%)	15	25	40	20	100
得分	**9**	**25**	**25**	**10**	**100**

系　　别：计算机科学与技术

专　　业：计算机科学与技术

班级姓名：计类 1707　张谦益

日　　期：

成　　绩：

同组成员：无

指导教师：丁濛

一、实验目的

1. 掌握二叉树的链式存储结构；
2. 验证二叉树的前续、中序、后序（递归方式）以及层序的遍历；(验证)
3. 验证二叉树的中序遍历（非递归方式的实现以及不用栈的实现）；(验证)
4. 验证二叉搜索树的插入、查找操作以及这些操作的复杂度和树高之间的关系；(验证)
5. 实现一个霍夫曼编码或实现一棵 AVL 树。（设计、综合）

二、实验内容

2.1 项目一

实现以链表形式存储的二叉树，要求具有如下功能：

1. 构造，析构；
2. 递归版的前序遍历、中序遍历、后序遍历；
3. 层序遍历；
4. 通过用户输入（层序完全二叉树形式），建立一棵二叉树。用户输入以字符串的形式给出，其中'#' 表示空节点。

2.2 项目二

改造第一题中的二叉树存储节点形式，使其支持能以非递归且不用额外栈的形式进行中序遍历。

2.3 项目三

在第二题的基础上，实现一棵二叉搜索树 (Binary Search Tree)，要求提供以下功能：

1. 向树中插入一个元素；
2. 前序、中序遍历一棵 BST；
3. 查找某个元素。

编写测试用例测试你的程序，验证 BST 树的查找和树高的关系。

2.4 项目四

实现一个霍夫曼编码 (在平台上完成)。(4, 5 任选一道即可)

1. 用户提供待编码的字符以及每个字符的频度，你给出编码结果。注意：构造霍夫曼树时，左儿子的频度应小于右儿子的频度；编码时，左儿子的前缀码为 0，右儿子的前缀码为 1。
2. 得到每个字符的编码后，输入一组由给定字符组成的明文，给出其对应的编码结果；
3. 然后对于编码结果，根据编码规则，写出其对应的明文。

2.5 项目五

实现一棵 AVL 树 (在平台上完成)。实现一棵 AVL 树，支持一个一个向树中加入元素，前序以及中序遍历 AVL 树的功能；支持查找 AVL 是否包含某个元素的功能。测试你的 AVL 树和 BST 树，在同样数据下的查找效率。

三、 实验过程

3.1 项目 1

大致思想 blablabla

(1) 实验步骤

(2) 必要代码

```
//
// main.cpp
// address_demo
//
// Created by Ding Meng on 10/26/17.
// Copyright (c) 2017 Ding Meng. All rights reserved.
//
//#include <stdio.h>
//int main(int argc, const char * argv[]) {
//    int a[4] = {0};
//
//    printf("a:%lx\n", a);
//    printf("a+1:%lx\n", a + 1);
//
//    for(int i = 0; i < 4; ++i)
//        a[i] = i + 1;
//    printf("a+1:%lx\n", a + 1);
//    printf("&a[3]:%lx\n", &a[3]);
//    printf("(int)a + 1:%lx\n", (unsigned long int)a + 1);
//    *(int*)(( long int)a + 1) = 1;
//    printf("(int)a + 2:%lx\n", (unsigned long int)a + 2);
```

```
//    printf("a[0]:%d a[1]:%d\n", a[0], a[1]);
//
//    return 0;
//}
//


#include <stdio.h>
#include <stdlib.h>
//#include <malloc.h>
#include <iostream>
using namespace std;
/*************************************************/
#define OK    1
#define ERROR 0
#define TRUE  1
#define FALSE 0
typedef int   Status;
typedef int   ElemType;
/*************************************************/
typedef struct list {
    ElemType *data;               //数据域
    int length;              //指针域
    list() :data(NULL), length(0) {}//构造函数
    ~list() {//析构函数
        delete[]data;
        data = NULL;
        length = 0;
    }
}Node, *pNode;
/*************************************************/
//在表后添加一个新数据---O(1)
//向表后加上值为d的元素
void InputList1(pNode L, ElemType d)
{
    int *temp = new int[L->length + 1];//用int型指针开辟一个长度为length+1的空间
    for (int i = 0; i < L->length; i++)//将data中元素赋给temp
        temp[i] = L->data[i];
    temp[L->length] = d;     //将d加到最后面
    delete L->data;        //删除data
    L->data = temp;        //将data的指向temp
    L->length++;         //长度加1
}
/*************************************************/
//向表中特定位置插入一个新数据---O(n)
//将至为d的元素加到第idx个元素后面
void InputList2(pNode L, int idx, ElemType d) {

    if (idx < 0 || L->length == 0 || idx > L->length - 1 || L->length<0)
        return;
```

```
    //当idx<0|| L->length == 0 || idx > L->length - 1 || L->length<0时不进行任何操作
    int *q = new int[L->length + 1];//用int型指针开辟一个长度为length+1的空间
    for (int i = 0; i < idx; i++)//将data的前idx个元素赋给q
        q[i] = L->data[i];
    q[idx] = d;       //令q的第idx+1个元素的值为d
    for (int i = idx; i < L->length; i++)//将data的第idx个元素之后的元素赋给temp
        q[i + 1] = L->data[i];
    delete L->data;//删除data
    L->data = q;//令data指向q
    L->length++;//长度加1
}
/*************************************************/
//打印所有元素---O(n)
void printlist(pNode L) {
    if (L->length == 0)
        cout << "栈空!!\n";//若表中没有元素，输出栈空
    for (int i = 0; i < L->length; i++)
        cout << L->data[i] << " ";//输出表中元素
    cout << endl;
}
/*************************************************/
//删除表中的某个特定元素
void DelList(pNode L, int idx) {//删除表中元素
    if (L->length <= 0 || idx<0 || idx >= L->length)//判断是否有效
        return;
    int *p = new int[L->length - 1];
    for (int i = 0; i < idx; i++)
        p[i] = L->data[i];
    for (int i = idx + 1; i < L->length; i++)//将第i个位置之后的元素前移
        p[i - 1] = L->data[i];
    delete L->data;//删除元素
    L->data = p;
    L->length--;//线性表长度减一
}
/*************************************************/
//将另外一张表内的内容添加到一个表的后面---O(n)
//将Lb添加到La的后面，然后将其赋给Lc
void MergeList(pNode La, pNode Lb, pNode Lc) {
    Lc->data = new int[Lb->length + La->length];
    //用int型指针开辟一个长度为Lb->length+La->length的空间
    for (int i = 0; i < La->length; i++)//将La的数据域的每个值依次赋给Lc
        Lc->data[i] = La->data[i];
    for (int i = 0; i < Lb->length;
         i++)//将Lb的数据域的每个值依次赋给Lc的第La->length之后的值
        Lc->data[La->length + i] = Lb->data[i];
    Lc->length = La->length + Lb->length;//长度变为La，Lb的长度之和
}
/*************************************************/
//查找表中是否含有特定元素
void findlist(pNode L, ElemType d) {//d为特定元素的值
```

```
    if (L->length == 0) {//表为空
        cout << "没有该元素!!\n";
        return;
    }
    for (int i = 0; i < L->length; i++) {//遍历表判断是否有值为d的元素
        if (L->data[i] == d) {
            cout << "第" << i + 1 << "个元素是" << d << endl;
            return;
        }
    }
    cout << "没有该元素!!\n";
}
/************************************************/
int main()
{
    pNode La, Lb, Lc;//定义结构体变量，即表La，Lb，Lc
    La = new list;
    Lb = new list;
    Lc = new list;
    printf("The List of A:\n ");
    for (int i = 0; i < 10; i++) {
        int t=2*i;
        InputList1(La, t);
    }
    printlist(La);
    printf("\n");
    printf("The List of B:\n ");
    for (int i = 0; i<10; i++)
        InputList1(Lb, i);
    printlist(Lb);
    printf("\n");
    printf("Now Merge the List of A and B:\n ");
    MergeList(La, Lb, Lc);
    printlist(Lc);
    printf("\n \n");
    InputList2(La, 2, 3);
    printlist(La);
    DelList(La, 3);
    printlist(La);
    findlist(La, 5);
    return 0;
}
```

(3)　实验结果

3.2　项目 2

大致思想 blablabla sdfhasdfklj asd flas jkfjaksdf asd f fasdf aksdjf klasj dfa sdf asdf f

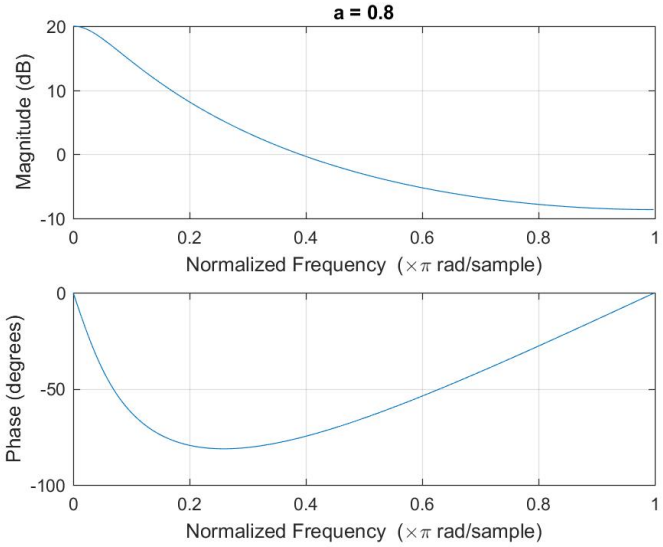

图 1: D 大调

(1) 实验步骤

(2) 必要代码

(3) 实验结果

3.3 项目 3

大致思想 blablabla

(1) 实验步骤

(2) 必要代码

(3) 实验结果

3.4 项目 4

四、实验总结