**全国计算机技术与软件专业技术资格（水平）考试**

**2016年上半年 程序员 下午试卷**

（考试时间 14:00～16:30 共 150 分钟）

|  |
| --- |
| **请按下述要求正确填写答题纸** |

1.在答题纸的指定位置填写你所在的省、自治区、直辖市、计划单列市的名称。

2.在答题纸的指定位置填写准考证号、出生年月日和姓名。

3.答题纸上除填写上述内容外只能写解答。

4.本试卷共 7 道题，试题一至试题四是必答题，试题五至试题六选答 1 道。每

题 15 分，满分 75 分。

5.解答时字迹务必清楚，字迹不清时，将不评分。

6.仿照下面例题，将解答写在答题纸的对应栏内。

**例题**

2016年上半年全国计算机技术与软件专业技术资格（水平）考试日期是（1）

月（2）日。

因为正确的解答是“5 月 20 日”，故在答题纸的对应栏内写上“5”和“20”

（参看下表）。

|  |  |
| --- | --- |
| 例题 | 解答栏 |
| （1） | 5 |
| （2） | 20 |

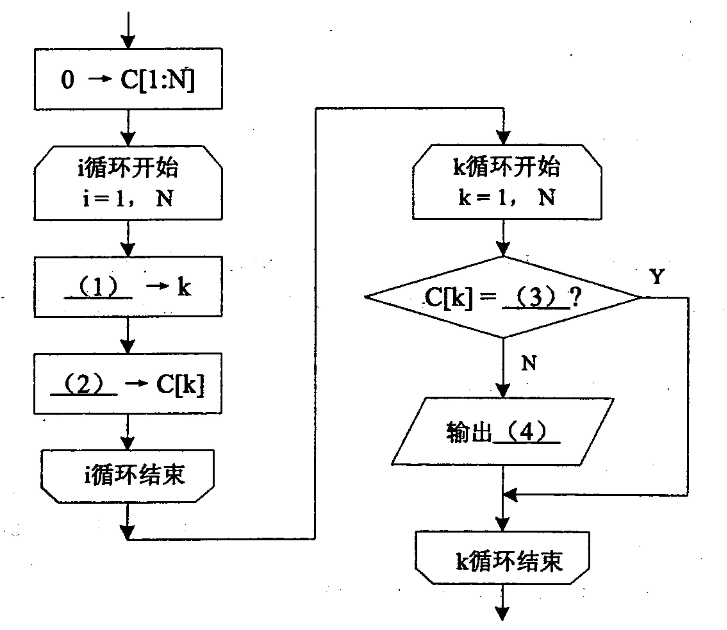
**试题一**

阅读以下说明和流程图，填补流程图和问题中的空缺(1)～(5)，将解答填入答题纸的对应栏内。

**【说明】**

设整型数组A[1：N]每个元素的值都是1到N之间的正整数。一般来说，其中会有一些元素的值是重复的，也有些数未出现在数组中。下面流程图的功能是查缺查重，即找出A[1：N]中所有缺的或重复的整数，并计算其出现的次数（出现次数为0时表示缺）。流程图中采用的算法思想是将数组A的下标与值看作是整数集[1：N]加上的一个映射，并用数组C[1：N]记录各整数出现的次数，需输出所有缺少的或重复的数及其出现的次数。

**【流程图】**



**【问题】**

如果数组A[1：5]的元素分别为{3，2，5，5，1}，则算法流程结束后输出结果为：

(5) 输出格式为：缺少或重复的元素，次数（0表示缺少）

**试题二**

阅读以下说明和C代码，填补代码中的空缺，将解答填入答题纸的对应栏内。

**【说明1】**

递归函数is\_elem(char ch, char \*set)的功能是判断ch中的字符是否在set表示的字符集合中，若是，则返回1，否则返回0。

**【C代码1】**

int is\_elem (char ch ,char\*set)

{

If(\*set==‘\0’)

return 0;

else

If((1))

return 1;

else

return is\_elem((2))

}

**【说明2】**

函数char\*combine(char\* setA,char \*setB)的功能是将字符集合A（元素互异，由setA表示）和字符集合B（元素互异，由setB表示）合并，并返回合并后的字符集合。

**【C代码2】**

char\*combine(char \*setA, char\*setB)

{

int i，lenA, lenB, lenC；

lenA=strlen(setA)；

lenB=strlen(setB)；

char\*setC=(char\*)malloc(lenA+lenB+1);

if(!setC)

return NULL;

strncpy(setC,setA,lenA)； //将setA的前lenA个字符复制后存入setC

lenC = (3);

for(i=0;i＜lenB；i++)

if((4)) //调用is\_elem判断字符是否在setA中

setC[lenC++]=setB[i];

(5) =‘/0’; //设置合并后字符集的结尾标识

return setC;

}

**试题三**

阅读以下说明和C代码，填补代码中的空缺，将解答填入答题纸的对应栏内。

**【说明】**

某文本文件中保存了若干个日期数据，格式如下（年／月／日）：

2005/12/1

2013/2/29

1997/10/11

1980/5/15

....

但是其中有些日期是非法的，例如2013/2/29是非法日期，闰年（即能被400整除或者能被4整除而不能被100整除的年份）的2月份有29天，2013年不是闰年。现要求将其中自1985/1/1开始、至2010/12/31结束的合法日期挑选出来并输出。

下面的C代码用于完成上述要求。

**【C代码】**

#include <stdio.h>

typedef struct{

int year, month, day；/\* 年，月，日\*/

}DATE；

int isLeap Year(int y) /\*判断y表示的年份是否为闰年，是则返回1，否则返回0\*/

{

return((y%4==0 && y%100!=0)Il(y%400==0))；

}

int isLegal(DATE date) /\*判断date表示的日期是否合法，是则返回1，否则返回0\*/

{

int y=date.year,m=date.month,d=date.day;

if(y<1985 II y>2010 II m<1 II m>12 II d<l II d>31) return 0;

if((m==4 ll m==6 ll m==9 II m==11)&&(1) ) return 0;

If(m==2){

if(isLeap Year(y) && (2)) return 1；

else

if(d28) return 0；

}

return 1;

}

Int Lteq(DATE d1，DATE d2)

/\*比较日期d1和d2，若d1在d2之前或相同则返回1，否则返回0\*/

{

Long t1,t2;

t1=d1.year\*10000+d1.month\*100+d1.day;

t2=d2.year\*10000+d2.month\*100+d2.day;

if((3)) return 1;

else return 0;

}

int main()

{

DATE date,start={1985,1,1},end={2010,12,30};

FILE\*fp;

fp=fopen(“d.txt”,”r”);

If((4))

return-1;

while(!feof(fp)){

if(fscanf(fp,”%d%d%d”,date.year,date.month,date.day)!=3)

break;

if((5)) /\*判断是否为非法日期\*/

continue；

if((6)) /\*调用Lteq判断是否在起至日期之间\*/

printf(“%d%d%d\n”,date.year,date.month,date.day);

}

fclose(fp);

Return 0;

}

**试题四**

阅读以下说明和C代码，填补代码中的空缺，将解答填入答题纸的对应栏内。

**【说明】**

二叉查找树又称为二叉排序树，它或者是一棵空树，或者是具有如下性质的二叉树。

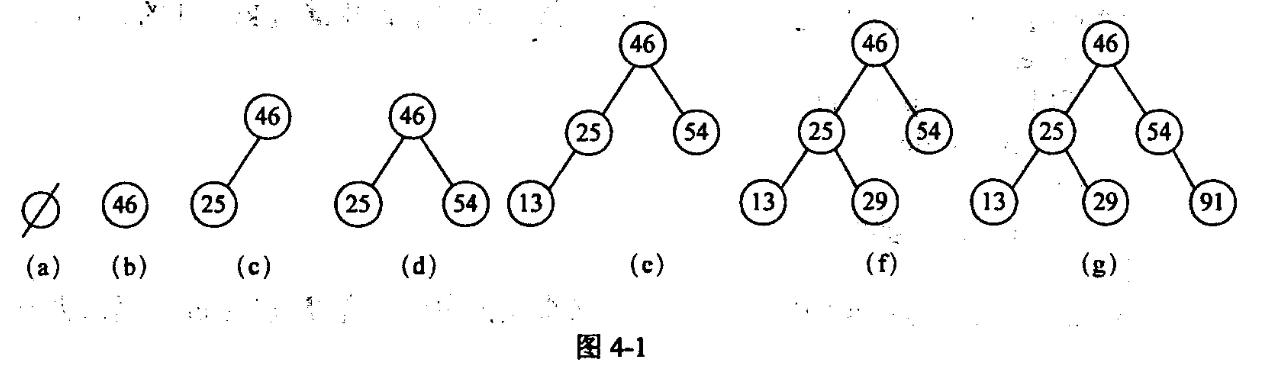
（1）若它的左子树非空，则左子树上所有结点的值均小于根结点的值。

（2）若它的右子树非空，则右子树上所有结点的值均大于根结点的值。

（3）左、右子树本身就是两棵二叉查找树。

二叉查找树是通过依次输入数据元素并把它们插入到二叉树的适当位置上构造起来的，具体的过程是：每读入一个元素，建立一个新结点，若二叉查找树非空，则将新结点的值与根结点的值相比较，如果小于根结点的值，则插入到左子树中，否则插入到右子树中；若二叉查找树为空，则新结点作为二叉查找树的根结点。

根据关键码序列{46，25，54，13，29，91}构造一个二叉查找树的过程如图4-1所示。



设二叉查找树采用二叉链表存储，结点类型定义如下：

typedef int KeyType；

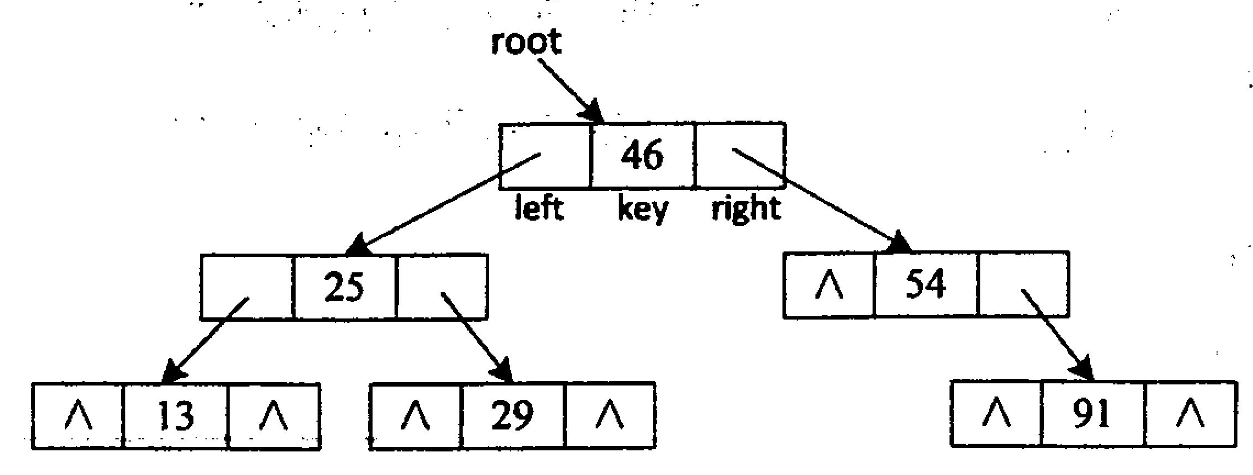
typedef struct BSTNode{

KeyType key；

struct BSTNode \*left,\*right；

}BSTNode,\*BSTree；

图4-1(g)所示二叉查找树的二叉链表表示如图4-2所示。



函数int InsertBST(BSTree \*rootptr,KeyType kword)功能是将关键码kword插入到由rootptr指示出根结点的二叉查找树中，若插入成功，函数返回1，否则返回0。

**【C代码】**

int lnsertBST(BSTree\*rootptr，KeyType kword)

/\*在二叉查找树中插入一个键值为kword的结点，若插入成功返回1，否则返回0；

\*rootptr为二叉查找树根结点的指针

\*/

{

BSTree p,father;

1. /\*将father初始化为空指针\*/

p=\*rootptr; /\*p指示二叉查找树的根节点\*/

while(p&&（2）){ /\*在二叉查找树中查找键值kword的结点\*/

father=p;

if(kword＜p-＞key)

p=p-＞left；

else

p=p-＞right；

}

if((3))return 0; /\*二叉查找树中已包含键值kword，插入失败\*/

p=(BSTree)malloc((4)); /\*创建新结点用来保存键值kword\*/

If(!p)return 0; /\*创建新结点失败\*/

p-＞key=kword；

p-＞left=NULL；

p-＞right=NULL；

If(!father)

(5) =p； /\*二叉查找树为空树时新结点作为树根插入\*/

else

if(kword＜father-＞key)

(6)； /\*作为左孩子结点插入\*/

else

(7)； /\*作右孩子结点插入\*/

return 1；

}/\*InsertBST\*/

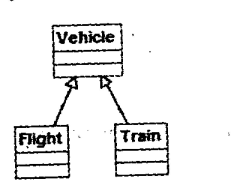
**试题五**

阅读以下说明和Java代码，填补代码中的空缺，将解答填入答题纸的对应栏内。

【说明】

以下Java代码实现两类交通工具（Flight和Train）的简单订票处理，

类Vehicle、Flight、Train之间的关系如图5-1所示。



**图5-1**

**【Java代码】**

import java.util.ArrayList;

import java.util.List;

abstract class Vehicle {

void book(int n) { //订 n张票

if (getTicket0() >=n ) {

decrease Ticket(n);

} else {

System.out.println(“余票不足！！“)；

}

}

abstract int getTicket();

abstract void decreaseTicket(int n);

};

class Flight（1）{

Private（2）tickets=216； //Flight的票数

Int getTicket(){

Return tickets；

}

void decreaseTicket(int n){

tickets=tickets - n；

}

}

class Train（3）{

Private（4）tickets=2016; //Train的票数

int getTicket() {

return tickets;

}

void decreaseticket(int n) {

tickets = tickets - n;

}

}

public class Test {

public static void main(String[] args) {

System.out.println（“欢迎订票!")；

ArrayListVehicle v = new ArrayListVehicle();

v.add(new Flight());

v.add(new Train());

v.add(new Flight());

v.add(new Train());

v.add(new Train());

for(int i=0;iv.size(); i++){

（5）（i+1）; //订i+1张票

System.out.println（“剩余票数：”+v.get(i).getTicket())；

}

}

}

运行该程序时输出如下：

欢迎订票！

剩余票数：215

剩余票数：2014

剩余票数：（6）

剩余票数：（7）

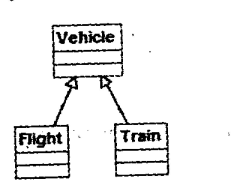
剩余票数：（8）

**试题六**

阅读下列说明和C++代码，填补代码中的空缺，将解答填入答题纸的对应栏内。

**【说明】**

以下C++代码实现两类交通工具（Flight和Train）的简单订票处理，类Vehicle、Flight、Train之间的关系如图6-1所示。



**图6-1**

**【C++代码】**

#include <iostream>

#include <vector>

using namespace std;

class Vehicle{

public:

virtual ~Vehicle(){}

void book(int n){ //订n张票

if(getTicket() >= n){

decreaseTicket(n);

} else{

coutn“余票不足！！”；

}

}

virtual int getTicket()=0;

virtual void decreaseTicket(int)=0；

}；

Class Flight: （1）{

private:

（2）tickets; //Flight的票数

public:

int getTicket();

void decreaseTicket(int);

};

class Train: （3）{

private:

（4）tickets; //Train的票数

public:

int getTicket();

void decreaseTicket(int);

};

int Train::tickets = 2016; //初始化Train的票数为2016

int Flight::tickets = 216; //初始化Flight的票数为216

int Train：：getTicket() { return tickets；}

void Train：：decreaseTicket(int n){ tickets=tickets - n;}

int Flight::getTicket(){return tickets; }

void Flight::decreaseTicket(int n) { tickets= tickets - n;}

int main() {

vector<Vehicle\*> v;

v.push\_back(new Flight());

v.push\_back(new Train());

v;push\_back(new Flight());

v.push\_back(new Tram());

v.push\_back(new Train());

cout《"欢迎订票！”《endl：

for (int i= 0;i<v.size(); i++) {

（5）(i+1)； //订i+l张票

cout《“剩余票数：”<<（\*V[i]）.getTicket()<<endl;

}

for(vector<Vehicle\*>：：iterator it = v.begin(); it != v.end(); it ++) {

if (NULL !=\*it) {

delete\*it;

\*it = NULL;

}

}

v.clear();

Return0;

}

运行该程序时输出如下：

欢迎订票！

剩余票数：215

剩余票数：2014

剩余票数：（6）

剩余票数：（7）

剩余票数：（8）