2016 年上半年程序员考试下午真题

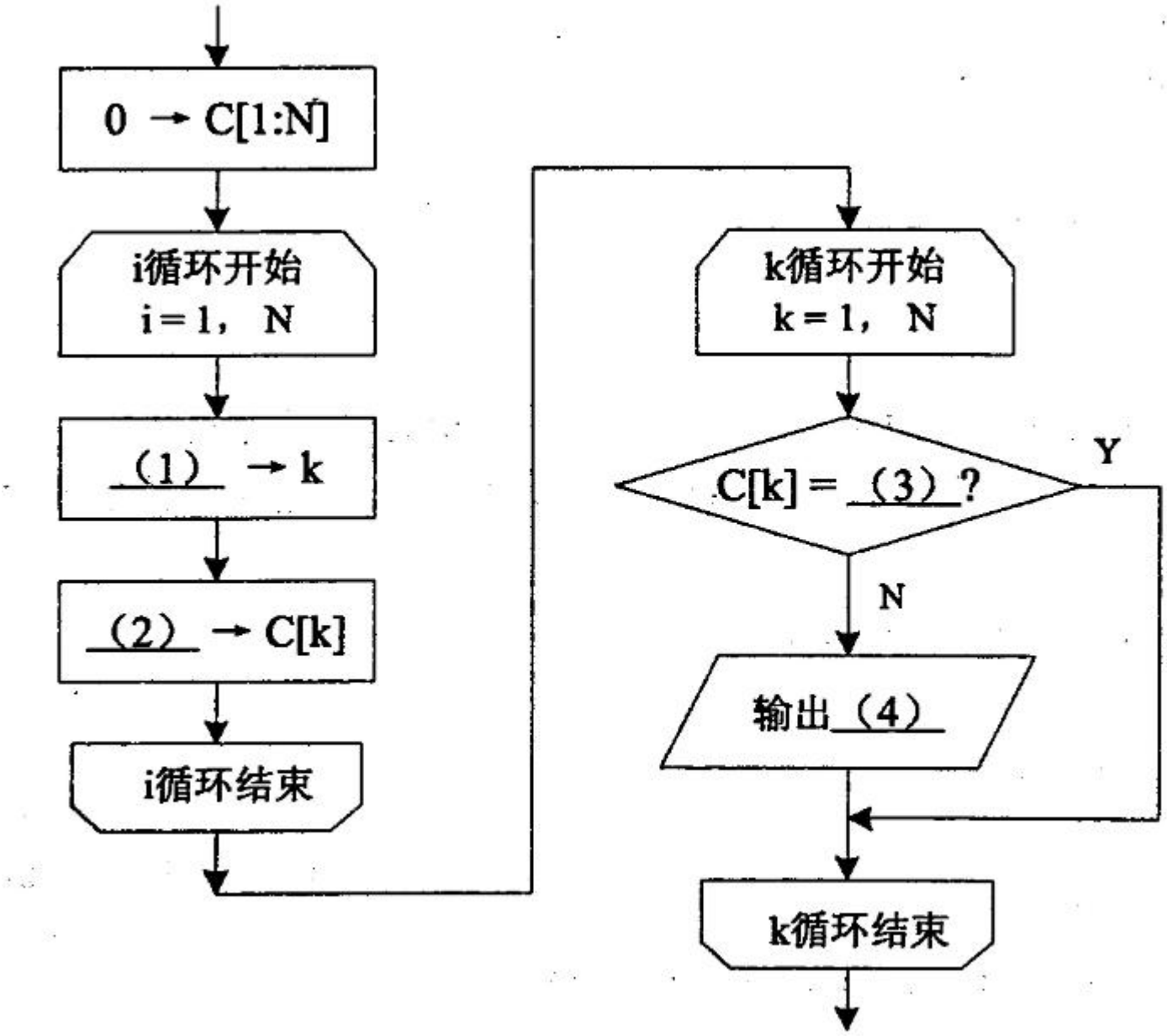
（参考答案）

● 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）

● 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;
}
```

● 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 (d>28) 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__(3)__
{
  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;
}
```

● 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。

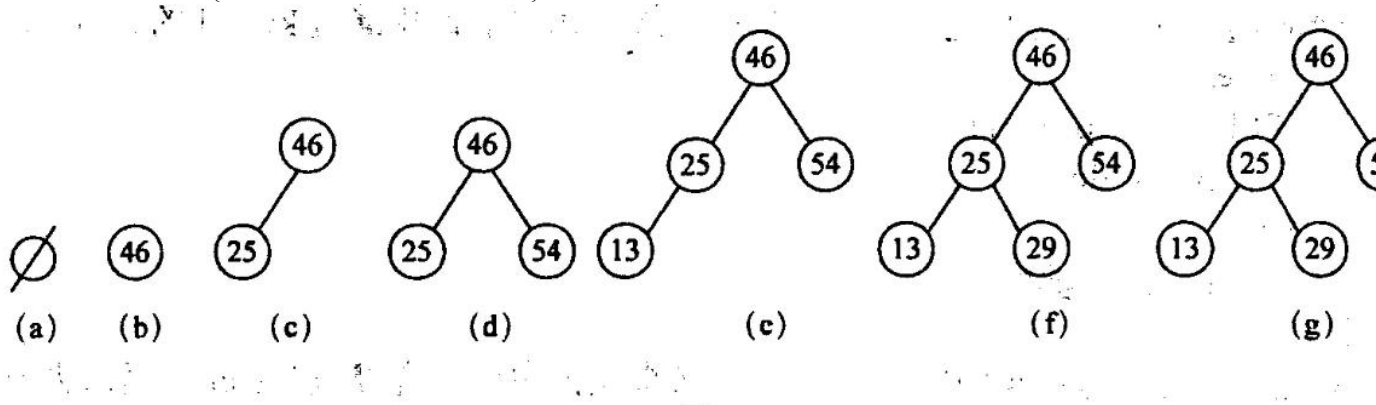
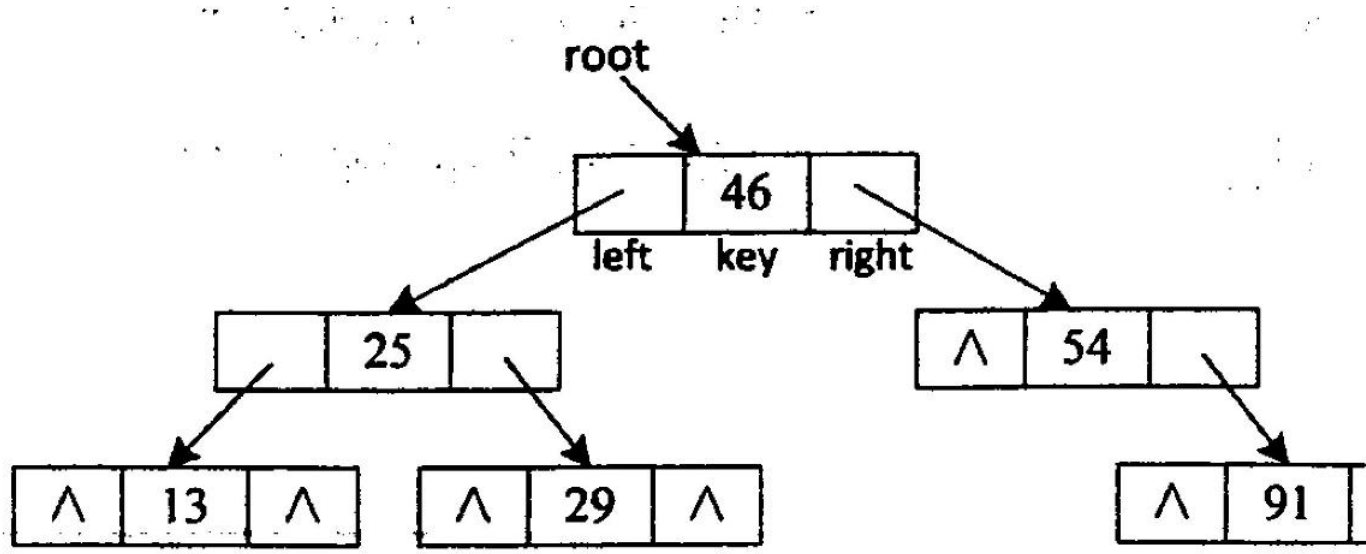


图 4-1

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*/
```

● 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。

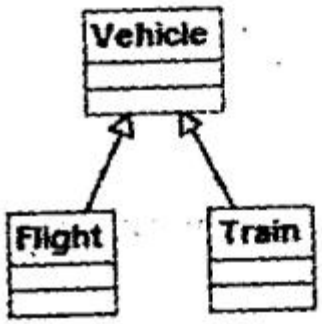


图 5-1

【Java 代码】

```
      import jav
（5）A．util.ArrayList;
      import jav
（6）A．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！");
        ArrayList<Vehicle> v = new ArrayList<Vehicle>();
        v.add(new Flight());
        v.add(new Train());
        v.add(new Flight());
        v.add(new Train());
        v.add(new Train());

         for (int i=0;i<v.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）

● 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。

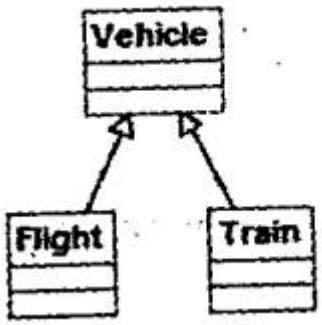


图 6-1

【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{
         cout<<n<<“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）