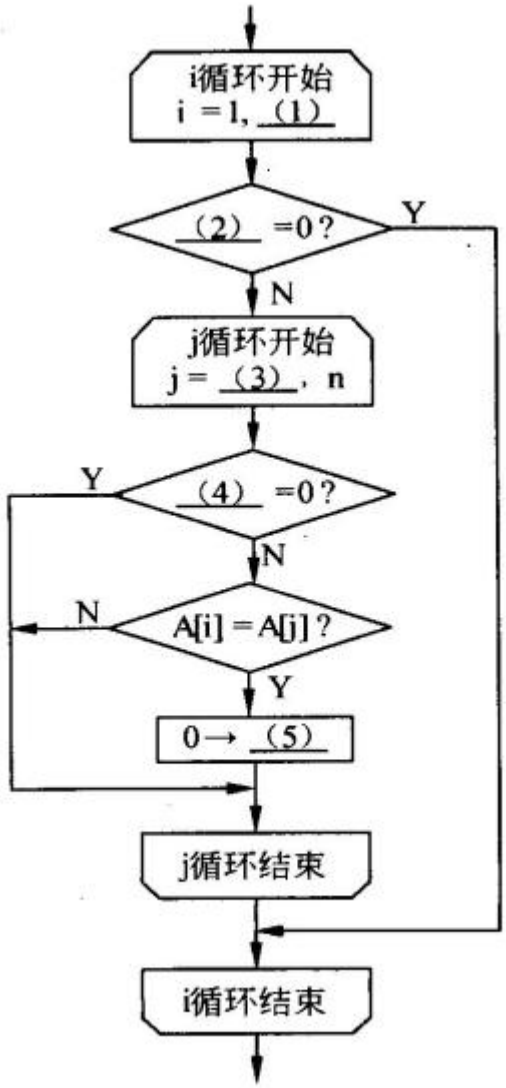
2012 年上半年程序员考试下午真题（参考答案）

● 阅读以下说明和流程图，填补流程图中的空缺（1）～（5），将解答填入答题纸的对应栏内。【说明】 已知数组 A[1:n]中各个元素的值都是非零整数，其中有些元素的值是相同的（重复）。为删除其中重复的值，可先通过以下流程图找出所有的重复值，并对所有重复值赋 0 标记。该流程图采用了双重循环。 处理思路：如果数组 A 某个元素的值在前面曾出现过，则该元素赋标记值 0。例如，假设数组 A 的各元素之值依次为 2，5，5，1，2，5，3，则经过该流程图处理后，各元素之值依次为 2，5，0，1，0，0，3。

【流程图】



● 阅读以下说明、C 程序代码和问题 1 至问题 3，将解答写在答题纸的对应栏内。【说明 1】 设在某 C 系统中为每个字待型数据分配 1 个字节，为每个整型（int）数据分配 4 个字节，为每个指针分配 4 个字节，sizeof(x)用于计算为 x 分配的字节数。【C 代码】 #include <stdio.h> #include <string.h> int main__(2)__ { int arr[5]={10, 20, 30}; char mystr[]="JustAtest\n"; char *ptr=mystr; printf("%d %d %d\n", sizeof(int), sizeof(unsigned int), strlen(arr)); printf("%d %d\n", sizeof(char), sizeof(mystr); printf("%d %d %d\n", sizeof(ptr), sizeof(*ptr), strlen(ptr)); return 0; }
【说明 2】 const 是 C 语言的一个关键字，可以用来定义“只读”型变量。
【问题 1】（8 分）
请写出以上 C 代码的运行结果。
【问题 2】（4 分）
（1）请定义一个“只读”型的整型常量 size，并将其值初始化为 10；
（2）请定义一个指向整型变量 a 的指针 ptr，使得 ptr 的值不能修改，而 ptr 所指向的目标变量的值可以修改（即可以通过 ptr 间接修改整型变量 a 的值）。
注：无需给出整型变量 a 的定义。

【问题 3】（3 分）

某 C 程序文件中定义的函数 f 如下所示，请简要说明其中 static 的作用，以及形参表“const int arr[]”中 const 的作用。

```
static int f(const int arr[])
{
 /* 函数体内的语句省略 */
}
```

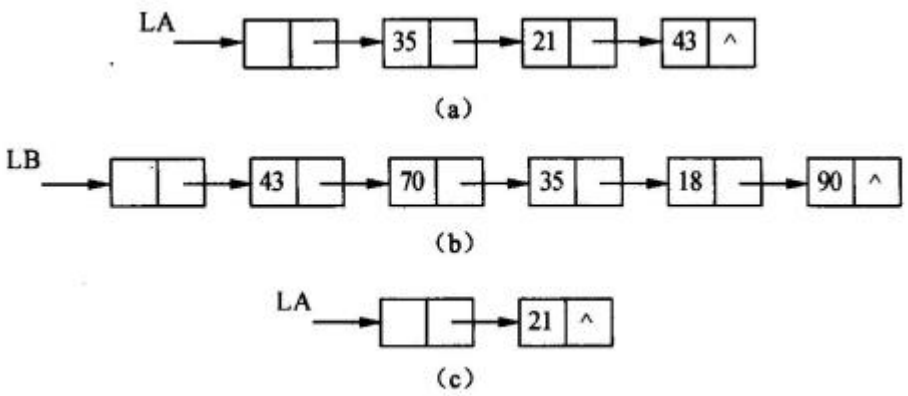
● 阅读以下说明和 C 函数，填补 C 函数中的空缺（1）～（6），将解答写在答题纸的对应栏内。【说明】 函数 numberOfwords(char message[])的功能是计算存储在 message 字符数组中的一段英文语句中的单词数目，输出每个单词（单词长度超过 20 时仅输出其前 20 个字母），并计算每个英文字母出现的次数（即频数），字母计数时不区分大小写。 假设英文语句中的单词合乎规范（此处不考虑单词的正确性），单词不缩写或省略，即不会出现类似 don't 形式的词，单词之后都为空格或标点符号。 函数中判定单词的规则是： （1）一个英文字母串是单词； （2）一个数字串是单词； （3）表示名词所有格的撇号（'）与对应的单词看作是一个单词。 除上述规则外，其他情况概不考虑。 例如，句子“The 1990's witnessed many changes in people's concepts of conservation.”中有 10 个单词，输出如下： The 1990's witnessed many changes in people's concepts of conservation 函数 numberOfwords 中用到的部分标准库函数如下表所述。

函 数 原 型	说 明
int islower(int ch);	若 ch 表示一个小写英文字母，则返回一个非 0 整数，否则返回 0
int isupper(int ch);	若 ch 表示一个大写英文字母，则返回一个非 0 整数，否则返回 0
int isalnum(int ch);	若 ch 表示一个英文字母或数字字符，则返回一个非 0 整数，否则返回 0
int isalpha (int ch);	若 ch 表示一个英文字母，则返回一个非 0 整数，否则返回 0
int isdigit (int ch);	若 ch 表示一个数字字符，则返回一个非 0 整数，否则返回 0

【C 函数】

```
int numberOfwords (char message[]) { char wordbuffer[21], i=0;  /* i 用作 wordbuffer 的下标 */  （1） pstr;  int ps[26]={0};  /* ps[0]用于表示字母'A'或'a'的频数 */ /* ps[1]用于表示字母'B'或'b'的频数，依此类推 */  int wordcounter=0;  pstr=message;  while (*pstr) {  if(（2）(*pstr)) {/* 调用函数判定是否为一个单词的开头字符 */  i=0;  do{/* 将一个单词的字符逐个存入 wordbuffer[]，并进行字母计数 */  wordbuffer[i++]=*pstr;  if (isalpha (*pstr)) {  if(（3）(*pstr)) ps[*pstr-'a']++;  else ps[*pstr-'A']++;  }  （4）;  /* pstr 指向下一字符 */  }while (i<20 && (isalnum(*pstr)||*pstr=='\')); if (i>=20) /* 处理超长单词（含名词所有格形式） */  while (isalnum (*pstr)||*pstr=='\') { pstr++; }  （5）='\0';  /* 设置暂存在 wordbuffer 中的单词结尾 */ wordcounter++;  /* 单词计数 */  puts (wordbuffer);  /* 输出单词 */  }  （6）;  /* pstr 指向下一字符 */  }  return wordcounter; }
```

● 阅读以下说明和 C 函数，填补 C 函数中的空缺（1）～（5），将解答写在答题纸的对应栏内。【说明】 函数 SetDiff(LA,LB)的功能是将 LA 与 LB 中的共有元素从 LA 中删除，使得 LA 中仅保留与 LB 不同的元素，而 LB 不变，LA 和 LB 为含头结点的单链表的头指针。 例如，单链表 LA、LB 的示例如下图中的（a）、（b）所示，删除与 LB 共有的元素后的 LA 如下图中的（c）所



示。 链表的结点类型定义如下： typedef struct Node { int data; struct Node *next; }Node, *LinkList; 函数 SetDiff(LinkList LA, LinkList LB)的处理思路如下： （1）从 LA 的第一个元素结点开始，令 LA 的第一个元素为当前元素。 （2）在 LB 中进行顺序查找，查找与 LA 的当前元素相同者，方法是令 LA 的当前元素先与 LB 的第一个元素进行比较，若相等，则结束在 LB 中的查找过程，否则继续与 LB 的下一个元素比较，重复以上过程，直到 LB 中的某一个元素与 LA 的当前元素相等（表明查找成功），或者到达 LB 的表尾（表明查找失败）为止。 （3）结束在 LB 表的一次查找后，若在 LB 中发现了与 LA 的当前元素相同者，则删除 LA 的当前元素，否则保留 LA 的当前元素。 （4）取 LA 的下一个元素为当前元素，重复（2）、（3），直到 LA 的表尾。【C 函数】 void SetDiff (LinkList LA, LinkList LB) { LinkList pre, pa, pb; /* pa 用于指向单链表 LA 的当前元素结点，pre 指向 pa 所指元素的前驱 */ /* pb 用于指向单链表 LB 的元素结点 */ __（1）__; /* 开始时令 pa 指向 LA 的第一个元素 */ pre=LA; while (pa) { pb=LB->next; /* 在 LB 中查找与 LA 的当前元素相同者，直到找到或者到达表尾 */ while(__（2）__) { if (pa->data==pb->data) break; __（3）__; } if (!pb) { /* 若在 LB 中没有找到与 LA 中当前元素相同者，则继续考察 LA 的后续元素 */ pre=pa; pa=pa->next; } else{ /* 若在 LB 中找到与 LA 的当前元素相同者，则删除 LA 的当前元素 */ pre->next=__（4）__; free (pa); pa=__（5）__; } } }

● 阅读以下说明和 C++代码，填补 C++代码中的空缺（1）～（6），将解答写在答题纸的对应栏内。【说明】 已知某公司按周给员工发放工资，其工资系统需记录每名员工的员工号、姓名、工资等信息。其中一些员工是正式的，按年薪分周发放（每年按 52 周计算）；另一些员工是计时工，以小时工资为基准，按每周工作小时数核算发放。 下面是实现该工资系统的 C++代码，其中定义了四个类：工资系统类 PayRoll，员工类 Employee，正式工类 Salaried 和计时工类 Hourly，Salaried 和 Hourly 是 Employee 的子类。【C++代码】 //头文件和域名空间略 const int EMPLOYEE_NUM=5; class Employee { protected: int empCode; //员工号 string name; //员工姓名 double salary; //周发放工资 public: Employee(const int empCode, const string &name) { this->empCode=empCode; this->name=name; } virtual ~Employee__(5)__{ } virtual void pay__(6)__=0; double getSalary__(7)__ { return this->salary; } }; class Salaried __（1）__ { private: double payRate; //年薪 public: Salaried(const int empCode, const string &namet double payRate):Employee (empCode, name) { this->payRate=payRate; } void pay__(8)__ { this->salary=__（2）__; //计算正式员工的周发放工资数 cout << this->name << ":" << this->salary<<endl; }; class Hourly __（3）__ { private: double payRate; //小时工资数 int hours; //周工作小时数 public: Hourly (const int empCode. const string &name, int hours, double payRate) :Employee(empCode, name) { this->payRate=payRate; this->hours=hours; } void pay__(9)__ { this->salary=__（4）__; //计算计时工的周发放工资

数 cout << this->name << ":" << this->salary << endl; } }; class PayRoll { public: void pay (Employee*e[]) (for (int i=0; i<EMPLOYEE_NUM; i++) { e[i]->pay__(10)__; } } }; int main__(11)__ { PayRoll payRoll=new PayRoll; __（5）__ employees [EMPLOYEE_NUM]= { new Salaried (1001, "Zhang San", 58000.00), //此处省略对其他职工对象的生成 new Hourly (1005, "Li", 12, 50.00), }; payRoll->pay(__（6）__); double total=0.0; for (int i=0; i<EMPLOYEE_NUM; i++) { total+=employees[i]->getSalary__(12)__； } //统计周发放工资患额 cout<<"总发放额="<<total<<endl; delete payRoll; return 0; }

● 阅读以下说明和 Java 代码，填补 Java 代码中的空缺（1）～（6），将解答写在答题纸的对应栏内。【说明】 已知某公司按周给员工发放工资，其工资系统需记录每名员工的员工号、姓名、工资等信息。其中一些员工是正式的，按年薪分周发放（每年按 52 周计算）；另一些员工是计时工，以小时工资为基准，按每周工作小时数核算发放。 下面是实现该工资系统的 Java 代码，其中定义了四个类：工资系统类 PayRoll，员工类 Employee，正式工类 Salaried 和计时工类 Hourly，Salaried 和 Hourly 是 Employee 的子类。【Java 代码】 abstract class Employee { protected String name; //员工姓名 protected int empCode; //员工号 protected double salary; //周发放工资 public Employee(int empCode, String name) { this.empCode=empCode; this.name=name, } public double getSalary__(6)__{ return thissalary; } public abstract void pay__(7)__ ; } class Salaried __（1）__ Employee { private double annualSalary; Salaried(int empCode, String name, double payRate) { super(empCode, name) ; this.annualSalary=payRate; } public void pay __(8)__ { salary=__（2）__; //计算正式员工的周发放工资数 System.out.println(this.name+":"+this.salary); } } class Hourly __（3）__ Employee { private double hourlyPayRate; private int hours; Hourly(int empCode, String name, int hours, double payRate) { super(empCode, name); this.hourlyPayRate=payRate; this.hours=hours; } public void pay __(9)__ { salary=__（4）__; //计算计时工的周发放工资数 System.out.println(thisname+":"+this.salary); } } public class PayRoll { private __（5）__ employees[]={ new Salaried(1001, "Zhang San", 58000.00), //此处省略对其他职工对象的生成 new Hourly(1005, "Li", 12, 50.00) }; public void pay(Employee e[]) { for (int i=0; i<e.length; i++) { e[i]pay__(10)__； } } public static void main(String[] args) { PayRoll payRoll=new PayRoll__(11)__; payRoll.pay(__（6）__); double total=0.0; for (int i=0;i<payRoll.employees.length; i++){ //统计周发放工资总额 total+=payRoll.employees[i].getSalary__(12)__; } System.out.println(total); } }