

# 全国计算机技术与软件专业技术资格（水平）考试

## 2009 年下半年 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 6 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。

试题号	一~四	五~六
选择方法	必答题	选答 1 题

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

2009 年下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“11 月 14 日”，故在答题纸的对应栏内写上“11”和“14”（参看下表）。

例题	解答栏
(1)	11
(2)	14

试题一（共 15 分）

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】

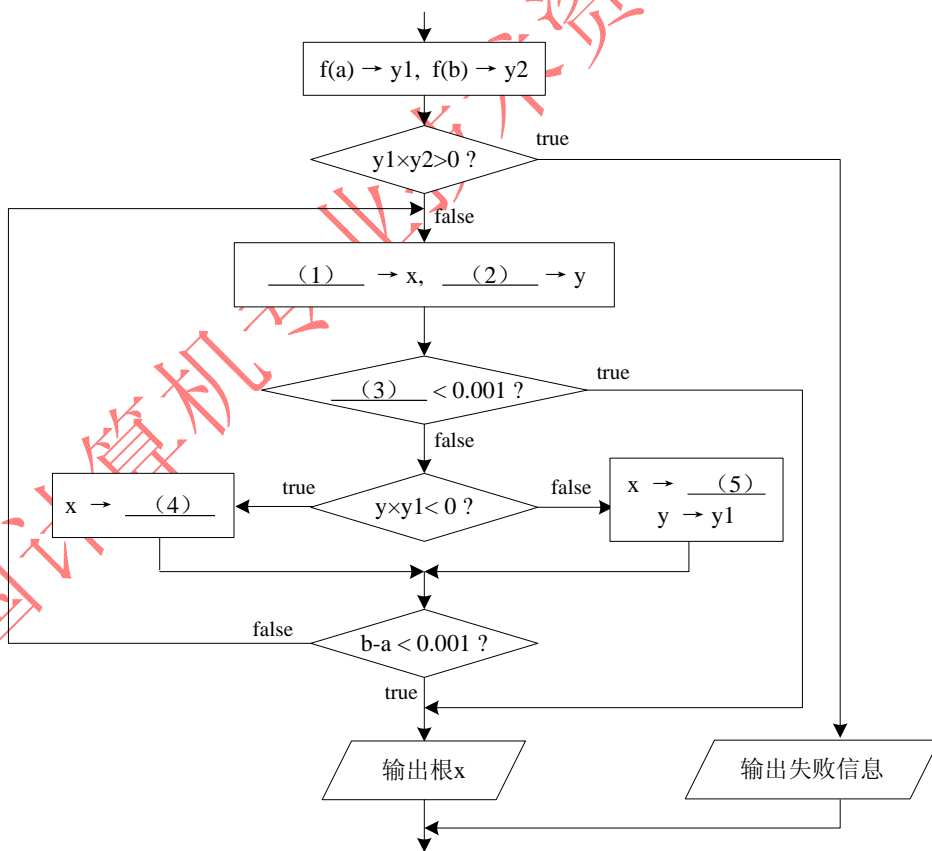
求连续函数  $f(x)$  的根（方程  $f(x)=0$  的解）的最简单方法是二分法。为此，首先需要在若干点上检查函数值的符号，如果发现  $f(a)$  与  $f(b)$  符号相反（ $a < b$ ），则在区间  $(a, b)$  中必然存在  $f(x)$  的根。因为当  $x$  从  $a$  变到  $b$  时，连续函数的值将从正变到负（或从负变到正），必然要经过 0。区间  $(a, b)$  就是根的初始范围。

取该区间的中点  $m$ ，如果  $f(m)=0$ ，则根就是  $m$ 。如果  $f(a)$  与  $f(m)$  符号相反，则根一定在区间  $(a, m)$  中；如果  $f(m)$  与  $f(b)$  符号相反，则根一定在区间  $(m, b)$  中。因此，根的范围缩小了一半。

依此类推，将区间一半一半地分下去，当区间的长度很小（达到根的精度要求，例如 0.001）时，或者当区间中点处的函数值几乎接近于 0（即绝对值小于预先规定的微小量，例如 0.001）时，近似计算就可以结束了。

以下流程图描述了用二分法近似计算区间  $(a, b)$  中  $f(x)$  的根的过程。

【流程图】



试题二（共 15 分）

阅读以下说明和C函数，将应填入 (n) 处的字句写在答题纸的对应栏内。

【说明 1】

函数 Counter(int n, int w[])的功能是计算整数 n 的二进制表示形式中 1 的个数，同时用数组 w 记录该二进制数中 1 所在位置的权。

例如，十进制数 22 的二进制表示为 10110。对于该二进制数，1 的个数为 3，在 w[0] 中存入 2（即  $2^1$ ）、w[1] 中存入 4（即  $2^2$ ）、w[2] 中存入 16（即  $2^4$ ）。

【C 函数 1】

```
int Counter(int n, int w[])
{
    int i = 0, k = 1;
    while ( ___(1)___ ) {
        if (n % 2) w[i++] = k;
        n = n / 2; ___(2)___;
    }
    return i;
}
```

【说明 2】

函数 Smove(int A[], int n)的功能是将数组中所有的奇数都放到所有偶数之前。其过程为：设置数组元素下标索引 i（初值为 0）和 j（初值为 n-1），从数组的两端开始检查元素的奇偶性。若 A[i]、A[j]都是奇数，则从前往后找出一个偶数，再与 A[j]进行交换；若 A[i]、A[j]都是偶数，则从后往前找出一个奇数，再与 A[i]进行交换；若 A[i]是偶数而 A[j]是奇数，则交换两者，直到将所有的奇数都排在所有偶数之前为止。

【C 函数 2】

```
void Smove(int A[], int n)
{
    int temp, i = 0, j = n-1;
    if (n < 2) return;
    while (i < j) {
        if (A[i] % 2 == 1 && A[j] % 2 == 1) { ___(3)___; }
        else if (A[i] % 2 == 0 && A[j] % 2 == 0) { ___(4)___; }
        else {
            if (___(5)___) {
                temp = A[i]; A[i] = A[j]; A[j] = temp;
            }
            i++, j--;
        }
    }
}
```

**试题三（共 15 分）**

阅读以下说明、C 函数和问题，将解答写入答题纸的对应栏内。

**【说明 1】**

函数 test\_f1(int m, int n)对整数 m、n 进行某种运算后返回一个整数值。

**【C 函数 1】**

```
int test_f1(int m, int n)
{
    int k;
    k = m > n ? m : n;
    for(;(k%m!=0) || (k%n!=0);k++);
    return k;
}
```

**【问题 1】（5 分）**

- (1) 请写出发生函数调用 test\_f1(9,6)时，函数的返回值；
- (2) 请说明函数 test\_f1 的功能。

**【说明 2】**

设在某 C 系统中为每个字符分配 1 个字节，为每个指针分配 4 个字节，sizeof(x)计算为 x 分配的字节数。

函数 test\_f2()用于测试并输出该 C 系统为某些数据分配的字节数。

**【C 函数 2】**

```
void test_f2()
{
    char str[] = "NewWorld";    char *p = str;    char i = '\0';
    void *ptr = malloc(50);

    printf("%d\t", sizeof(str));    printf("%d\n", sizeof(p));
    printf("%d\t", sizeof(i));    printf("%d\n ", sizeof(ptr));
}
```

**【问题 2】（4 分）**

请写出函数 test\_f2()的运行结果。

**【说明 3】**

函数 test\_f3(char s[])的功能是：将给定字符串 s 中的所有空格字符删除后形成的串保存在字符数组 tstr 中（串 s 的内容不变），并返回结果串的首地址。

**【C 函数 3】**

```
char *test_f3 (const char s[])
{
    char tstr[50]={'\0'};    unsigned int i, k = 0;
    for(i=0; i<strlen(s); i++)
        if (s[i] != ' ')    tstr[k++] = s[i];
    return tstr;
}
```

**【问题 3】（6 分）**

函数 test\_f3()对返回值的处理有缺陷，请指出该缺陷并说明修改方法。

#### 试题四（共 15 分）

阅读以下说明和 C 函数，将解答填入答题纸的对应栏内。

##### 【说明】

函数 del\_substr(S,T)的功能是从头至尾扫描字符串 S，删除其中与字符串 T 相同的子串，其处理过程为：首先从串 S 的第一个字符开始查找子串 T，若找到，则将后面的字符向前移动将子串 T 覆盖掉，然后继续查找子串 T，否则从串 S 的第二个字符开始查找，依此类推，重复该过程，直到串 S 的结尾为止。该函数中字符串的存储类型 SString 定义如下：

```
typedef struct {
    char *ch;          /*串空间的首地址*/
    int length;       /*串长*/
}SString;
```

##### 【C 函数】

```
void del_substr(SString *S, SString T)
{
    int i, j;
    if ( S->length < 1 || T.length < 1 || S->length < T.length )
        return;
    i = 0; /* i 为串 S 中字符的下标 */
    for (;) {
        j = 0; /* j 为串 T 中字符的下标 */
        while ( i < S->length && j < T.length ) { /* 在串 S 中查找与 T 相同的子串 */
            if ( S->ch[i]==T.ch[j] ) {
                i++; j++;
            }
            else {
                i = ___(1)___; j = 0; /* i 值回退，为继续查找 T 做准备 */
            }
        }
        if ( ___(2)___ ) { /* 在 S 中找到与 T 相同的子串 */
            i = ___(3)___; /* 计算 S 中子串 T 的起始下标 */
            for(k = i+T.length; k<S->length; k++) /* 通过覆盖子串 T 进行删除 */
                S->ch[___(4)___] = S->ch[k];
            S->length = ___(5)___; /* 更新 S 的长度 */
        }
        else break; /* 串 S 中不存在子串 T */
    }
}
```

从下列 2 道试题（试题五至试题六）中任选 1 道解答。如  
果解答的试题数超过 1 道，则题号小的 1 道解答有效。

### 试题五（共 15 分）

阅读以下说明和 C++ 代码，将应填入 (n) 处的字句写在答题纸的对应栏内。

#### 【说明】

已知类 `LinkedList` 表示列表类，该类具有四个方法：`addElement()`、`lastElement()`、`numberOfElement()` 以及 `removeLastElement()`。四个方法的含义分别为：

`void addElement(Object)`: 在列表尾部添加一个对象；

`Object lastElement()`: 返回列表尾部对象；

`int numberOfElement()`: 返回列表中对象个数；

`void removeLastElement()`: 删除列表尾部的对象。

现需要借助 `LinkedList` 来实现一个 `Stack` 栈类，C++ 代码 1 和 C++ 代码 2 分别采用继承和组合的方式实现。

#### 【C++ 代码 1】

```
class Stack :public LinkedList{
public:
    void push(Object o){ addElement(o); }; //压栈
    Object peek(){ return (1); }; //获取栈顶元素
    bool isEmpty(){ //判断栈是否为空
        return numberOfElement() == 0;
    };
    Object pop(){ //弹栈
        Object o = lastElement();
        (2);
        return o;
    };
};
```

#### 【C++ 代码 2】

```
class Stack {
private:
    (3);
public:
    void push(Object o){ //压栈
        list.addElement(o);
    };
    Object peek(){ //获取栈顶元素
        return list.(4);
    };
};
```

---

```
};  
bool isEmpty(){ //判断栈是否为空  
    return list.numberOfElement() == 0;  
};  
Object pop(){//弹栈  
    Object o = list.lastElement();  
    list.removeLastElement();  
    return o;  
};  
};
```

**【问题】**

若类LinkedList新增了一个公有的方法removeElement(int index)，用于删除列表中第index个元素，则在用继承和组合两种实现栈类Stack的方式中，哪种方式下Stack对象可访问方法removeElement(int index)?     (5)     (A. 继承 B. 组合)

试题六（共 15 分）

阅读以下说明和Java代码，将应填入 (n) 处的字句写在答题纸的对应栏内。

【说明】

已知类 LinkedList 表示列表类，该类具有四个方法：addElement()、lastElement()、numberOfElement()以及 removeLastElement()。四个方法的含义分别为：

void addElement(Object): 在列表尾部添加一个对象；

Object lastElement(): 返回列表尾部对象；

int numberOfElement(): 返回列表中对象个数；

void removeLastElement(): 删除列表尾部的对象。

现需要借助 LinkedList 来实现一个 Stack 栈类，Java 代码 1 和 Java 代码 2 分别采用继承和组合的方式实现。

【Java 代码 1】

```
public class Stack extends LinkedList{
    public void push(Object o){ //压栈
        addElement(o);
    }
    public Object peek() { //获取栈顶元素
        return ____ (1) ____;
    }
    public boolean isEmpty(){ //判断栈是否为空
        return numberOfElement() == 0;
    }
    public Object pop(){ //弹栈
        Object o = lastElement();
        ____ (2) ____;
        return o;
    }
}
```

【Java 代码 2】

```
public class Stack {
    private ____ (3) ____;
    public Stack(){
        list = new LinkedList();
    }
    public void push(Object o){
        list.addElement(o);
    }
}
```

---

```
public Object peek(){//获取栈顶元素
    return list.__(4)_;
}
public boolean isEmpty(){//判断栈是否为空
    return list.numberOfElement() == 0;
}
public Object pop(){ //弹栈
    Object o = list.lastElement();
    list.removeLastElement();
    return o;
}
}
```

**【问题】**

若类LinkedList新增了一个公有的方法removeElement(int index)，用于删除列表中第index个元素，则在用继承和组合两种实现栈类Stack的方式中，哪种方式下Stack对象可访问方法removeElement(int index)? \_\_\_\_(5)\_\_(A. 继承 B. 组合)