2009 年上半年程序员考试下午真题（参考答案）

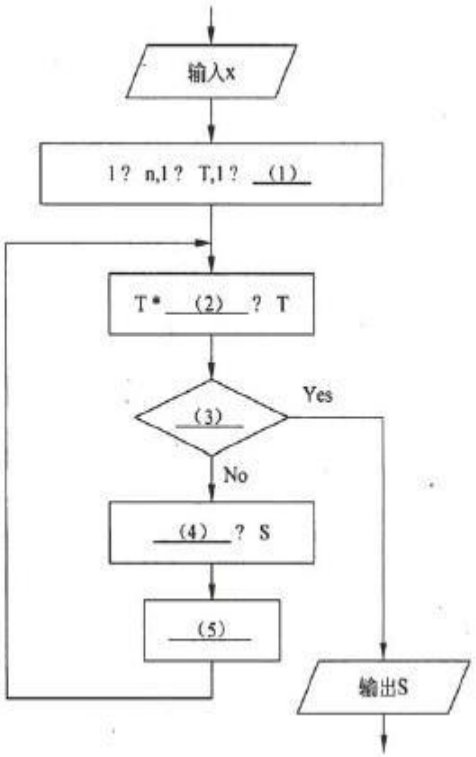
• **__(1)__（共 15 分）**

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】

下面的流程图采用公式 计算 的近似值。

设 x 位于区间 （0，1），该流程图的算法要点是逐步累积计算每项 $X^n/n!$的值 （作为 T），再逐步累加 T 值得到所需的结果 S。当 T 值小于 10^{-5} 时，结束计算。



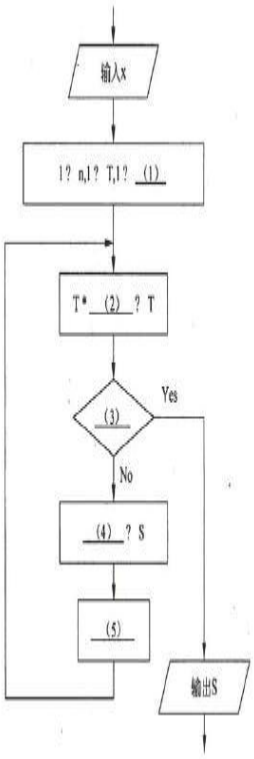
（）（共 15 分）

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】

下面的流程图采用公式 计算 的近似值。

设 x 位于区间 （0，1）， 该流程图的算法要点是逐步累积计算每项 $X^n/n!$的值 （作为 T），再逐步累加 T 值得到所需的结果 S。当 T 值小于 10^{-5} 时，结束计算。



- **__(2)__（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】

C 语言常用整型（int）或长整型（long）来说明需要处理的整数，在一般情况下可以满足表示及运算要求，而在某些情况下，需要表示及运算的整数比较大，即使采用更长的整型（例如，long long 类型，某些 C 系统会提供）也无法正确表示，此时可用一维数组来表示一个整数。

假设下面要处理的大整数均为正数，将其从低位到高位每 4 位一组进行分组（最后一组可能不足 4 位），每组作为 1 个整数存入数组。例如，大整数 2543698845679015847 在数组 A 中的表示如下（特别引入-1 表示分组结束）：

A[n]		A[6]	A[5]	A[4]	A[3]	A[2]	A[1]	A[0]
	...		-1	254	3698	8456	7901	5847

在上述表示机制下，函数 add_large_number(A，B，C)将保存在一维整型数组 A 和 B 中的两个大整数进行相加，结果（和数）保存在一维整型数组 C 中。

【C 函数】

```
void add_large_number(int A[]， int B[]， int C[])
{
  int i， cf ;  /*cf 存放进位*/
int t， *p;  /*t 为临时变量，p 为临时指针*/

cf = （1） ;
  for(i = 0; A[i]>-1 && B[i]>-1; i++) {
/*将数组
（2）A．B 对应分组中的两个整数进行相加*/
  t = （2） ;
  C[i] = t % 10000;
  cf = （3） ;
}

if ( （4） )  p = B;
else  p = A;
for( ; p[i]>-1; i++) {  /*将分组多的其余各组整数带进位复制入数组 C*/
  C[i] = (p[i] + cf) %10000;  cf = (p[i] + cf) /10000;
}

if ( cf > 0 )  C[i++] = cf;
  （5） = -1;  /*标志"和数"的分组结束*/
}
```

- **__(3)__（共 15 分）**

阅读以下说明、C 函数和问题，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：

若它的左子树非空，则其左子树上所有结点的键值均小于根结点的键值；

若它的右子树非空，则其右子树上所有结点的键值均大于根结点的键值；

左、右子树本身就是二叉查找树。

设二叉查找树采用二叉链表存储结构，链表结点类型定义如下：

```
typedef struct BiTnode{
int   key_value;  /*结点的键值，为非负整数*/
  struct BiTnode *left，*right;  /*结点的左、右子树指针*/
}*BSTree;
```

函数 find_key(root， key)的功能是用递归方式在给定的二叉查找树（root 指向根结点）中查找键值为 key 的结点并返回结点的指针；若找不到，则返回空指针。

【函数】

```
BSTree find_key(BSTree root， int key)
{
if ( （1） )
return NULL;
else
if (key == root-> key_value)
```

```
 return （2）__ ;
   else if (key < root -> key_value)
 return （3）__ ;
   else
 return （4）__ ;
}
```

【问题 1】

请将函数 find_key 中应填入（1）～（4）处的字句写在答题纸的对应栏内。

【问题 2】

若某二叉查找树中有 n 个结点，则查找一个给定关键字时，需要比较的结点个数取决于__（5）__。

● __(4)__（共 15 分）

阅读以下两个说明、C 函数和问题，将解答写入答题纸的对应栏内。

【说明 1】

函数 main__(5)__的功能旨在对输入的一个正整数 n，计算 $1^2+2^2+3^2+...+n^2$，但是对该函数进行测试后没有得到期望的结果。

【C 函数 1】

行号	代码
1	void main( )
2	{ int k, n, sum;
3	printf("input an integer:");
4	scanf("%d", n);
5	for(k = 1; k<=n; k++);
6	sum += k*k;
7	printf("result: %d\n", sum);
8	}

1. 输入 5 测试上述 main 函数时，显示结果如下所示。

```
input an integer:5
result: -582598909
```

2. 将行号为 7 的代码修改为：printf("n = %d\nresult: %d\n"， n， sum); 并再次输入 5 测试 main 函数，显示结果如下所示。

```
input an integer:5
n = 2293632
result: -582598909
```

【问题 1】（9 分）

请给出上述 main 函数中需要修改的代码行号，并给出修改后的整行代码。 行号 修改后的整行代码

行号	修改后的整行代码

【说明 2】

函数 test_f2__(6)__编译时系统报告有错，修改后得到函数 f2_B__(7)__。对函数 f2_B__(8)__进行编译时顺利通过，在某些 C 系统中执行时却由于发生异常而不能正确结束。

【C 函数 2】

```
void test_f2( )
{   char str[] = "test string";
    int i;
    for(i = 0; i < 4; i++, str++ )
        *str = 'a';
}
```

```
void f2_B( )
{   char *str = "test string";
    int i;
    for(i = 0; i < 4; i++, str++ )
        *str = 'a';
}
```

【问题 2】（6 分）

（1）请指出函数 test_f2 中不能通过编译的表达式；

（2）请指出可能导致函数 f2_B 运行异常的表达式。

（）（共 15 分）

阅读以下两个说明、C 函数和问题，将解答写入答题纸的对应栏内。

【说明 1】

函数 main()的功能旨在对输入的一个正整数 n，计算 ，但是对该函数进行测试后没有得到期望的结果。

【C 函数 1】

行号	代码
1	void main( )
2	{ int k, n, sum;
3	printf("input an integer:");
4	scanf("%d", n);
5	for(k = 1; k<=n; k++);
6	sum += k*k;
7	printf("result: %d\n", sum);
8	}

1. 输入 5 测试上述 main 函数时，显示结果如下所示。

```
input an integer:5
result: -582598909
```

2. 将行号为 7 的代码修改为：printf("n = %d\nresult: %d\n"， n， sum); 并再次输入 5 测试 main 函数，显示结果如下所示。

```
input an integer:5
n = 2293632
result: -582598909
```

【问题 1】（9 分）

请给出上述 main 函数中需要修改的代码行号，并给出修改后的整行代码。 行号 修改后的整行代码

行号	修改后的整行代码

【说明 2】

函数 test_f2()编译时系统报告有错，修改后得到函数 f2_B()。对函数 f2_B()进行编译时顺利通过，在某些 C 系统中执行时却由于发生异常而不能正确结束。

【C 函数 2】

```
void test_f2( )
{    char str[] = "test string";
     int i;
     for(i = 0; i < 4; i++, str++ )
          *str = 'a';
}
```

```
void f2_B( )
{    char *str = "test string";
     int i;
     for(i = 0; i < 4; i++, str++ )
          *str = 'a';
}
```

【问题 2】（6 分）

（1）请指出函数 test_f2 中不能通过编译的表达式；

（2）请指出可能导致函数 f2_B 运行异常的表达式。

● __(5)__ **（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】

C++标准模板库中提供了 map 模板类，该模板类可以表示多个“键-值”对的集合，其中键的作用与普通数组中的索引相当，而值用作待存储和检索的数据。此外，C++模板库还提供了 pair 模板类，该类可以表示一个“键-值”对。pair 对象包含两个属性：first 和 second，其中 first 表示“键-值”中的“键”，而 second 表示“键-值”中的“值”。

map 类提供了 insert 方法和 find 方法，用于插入和查找信息。应用时，将一个 pair 对象插入（insert）到 map 对象后，根据“键”在 map 对象中进行查找（find），即可获得一个指向 pair 对象的迭代器。

下面的 C++代码中使用了 map 和 pair 模板类，将编号为 1001、1002、1003 的员工信息插入到 map 对象中，然后输入一个指定的员工编号，通过员工编号来获取员工的基本信息。员工编号为整型编码，员工的基本信息定义为类 employee。

map 对象与员工对象之间的关系及存储结构如图 5-1 所示。

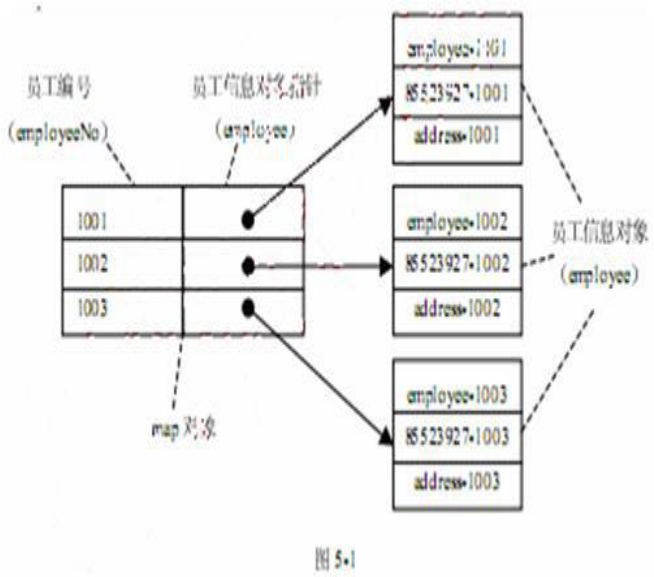


图 5-1

【C++代码】

```
#include <iostream>
#include <map>
#include <string>
using namespace std ;
class employee{
 （1）:
  employee(string name， string phoneNumber， string address){
    this->name = name;
    this->phoneNumber = phoneNumber;
    this->address = address;
  }
  string name;
  string phoneNumber;
  string address;
};
int main( )
{
  map <int， employee*> employeeMap;
  typedef pair <int， employee*> employeePair;
  for (int employIndex = 1001; employIndex <= 1003; employIndex++){
    char temp[10] ; //临时存储空间
    _itoa(employIndex，temp，10); //将 employIndex 转化为字符串存储在 temp 中
    string tmp(（2）); //通过 temp 构造 string 对象
    employeeMap. （3）( employeePair ( employIndex，
  new employee("employee-" + tmp，
     "85523927-"+tmp，
       "address-"+tmp)
     )
     ); //将员工编号和员工信息插入到 employeeMap 对象中
  }
```

```
 int employeeNo = 0;
 cout << "请输入员工编号:";
 __（4）__>> employeeNo;  //从标准输入获得员工编号
 map<int，employee*>::const_iterator it;
 it = __（5）__.find(employeeNo);  //根据员工编号查找员工信息
 if (it == employeeMap.end()) {
   cout << "该员工编号不存在 ！ " << endl;
   return -1;
 }
 cout << "你所查询的员工编号为：" << it->first << endl;
cout << "该员工姓名：" << it->second->name << endl;
 cout << "该员工电话：" << it->second->phoneNumber << endl;
 cout << "该员工地址：" << it->second->address << endl;
 return 0;
}
```

- **__(6)__（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】

java.util 包中提供了 HashMap 模板类，该模板类可以表示多个“键-值”对的集合，其中 “键” 的作用与普通数组中的索引相当，而 “值” 用作待存储和检索的数据。 HashMap 实现了 Map 接口。在 Map 接口中定义了 put 和 get 方法，put 方法表示 Map 对象中加入一个“键-值”对，get 方法则通过“键”来获取其对应的“值” 。

下面的 Java 代码中使用了 HashMap 模板类，将编号为 1001、1002、1003 的员工信息插入到 HashMap 对象中，然后输入一个指定的员工编号，通过员工编号来获取员工的基本信息。员工编号为整型编码，而员工的基本信息定义为类 employee。

HashMap 对象与员工对象之间的关系及存储结构如图 6-1 所示。

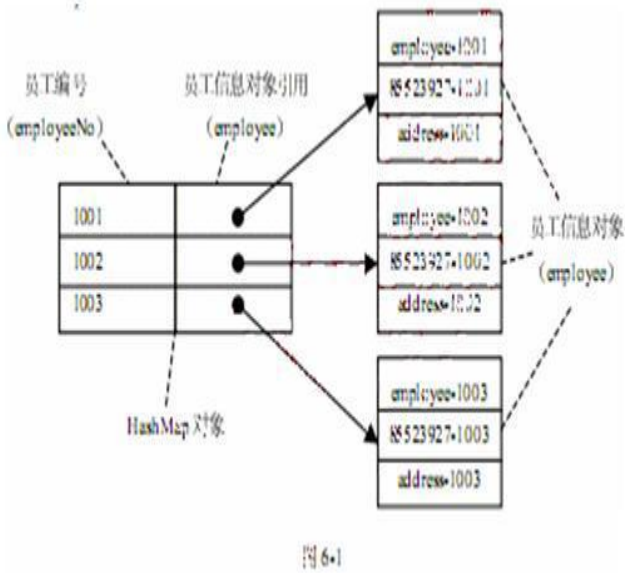


图 6-1

【Java 代码】

```
    import jav
（6）A．util.*;
```

```
class employee{
  employee(String name， String phoneNumber， String address){
    this.name = name;
    this.phoneNumber = phoneNumber;
    this.address = address;
  }
  String name;
  String phoneNumber;
  String address;
};
public class javaMain {

  public static void main(String[] args) {
  Map<Integer， employee> employeeMap = new HashMap<Integer， employee>();
  for (Integer employIndex = 1001; employIndex <= 1003; employIndex++){
    String tmp = employIndex. （1） ();
    employeeMap. （2） (employIndex， （3） ("employee-"+tmp，
    "85523927-"+tmp，
    "address-"+tmp
  )
    ); //将员工编号和员工信息插入到 employeeMap 对象中
  }

  int employeeNo = 0;
  System.out.print("请输入员工编号:");

     Scanner s= new Scanner(System.in);
  employeeNo = s.nextInt();   //从标准输入获得员工编号

  employee result = employeeMap.__（4）__(employeeNo);

  if (__（5）__== null)
  {
    System.out.println("该员工编号不存在 ！ ");
    return;
  }
  System.out.println("你所查询的员工编号为：" + employeeNo);
  System.out.println("该员工姓名：" + result.name);
  System.out.println("该员工电话：" + result.phoneNumber);
  System.out.println("该员工地址：" + result.address );
   }
}
```