2013 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。

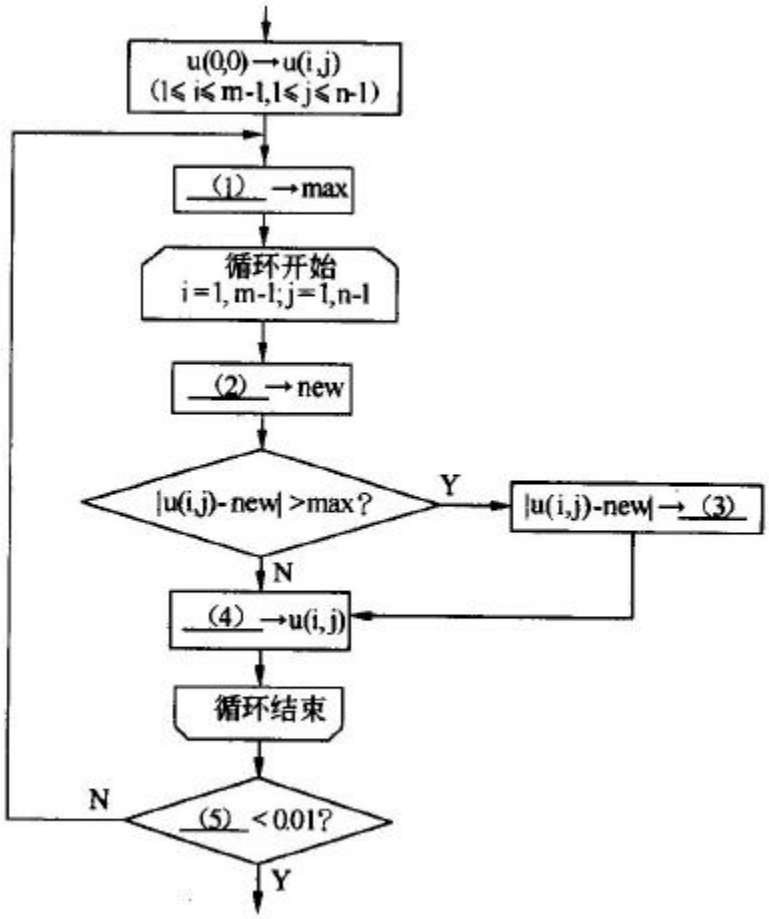
【说明】

平面上一个封闭区域内稳定的温度函数是一个调和函数。如果区域边界上各点的温度是已知的(非常数)，那么就可以用数值方法近似地计算出区域内各点的温度。

假设封闭区域是矩形，可将整个矩形用许多横竖线切分成比较细小的网格，并以最简单的方式建立坐标系统，从而可以将问题描述为:已知调和函数 u(i，j)在矩形 {0≤i≤m; 0≤j≤n}四边上的值，求函数 u 在矩形内部各个网格{(i，j)丨i=1，....m-1; j=1，.....，n-1}上的近似值。

根据调和函数的特点可以推导出近似算式:该矩形内任一网格点上的函数值等于其上下左右四个相邻网格点上函数值的算术平均值。这样，我们就可以用法代法来进行数值计算了。首先将该矩形内部所有网格点上的函数值设置为一个常数，例如 u(0,0); 然后通过该法代式计算矩形内各网格点上的新值。这样反复进行法代计算，若某次迭代后所 有的新值与原值之差别都小于预定的要求(如 0.01)，则结束求解过程。

【流程图】



● 阅读以下说明和 C 函数，填充函数中的空缺，将解答填入答题纸的对应栏内。

【说明】

函数 GetDateld(DATE date)的功能是计算并返回指定合法日期 date 是其所在年份的第几天。例如，date 表示 2008 年 1 月 25 日时，函数的返回值为 25，date 表示 2008 年 3 月 3 日时，函数返回值为 63。

函数 Kday_Date(int theyear，int k)的功能是计算并返回指定合法年份 theyear(theyear≥1900)的第 k 天(1≤k≤365)所对应的日期。例如，2008 年的第 60 天是 2008 年 2 月 29 日，2009 年的第 60 天是 2009 年 3 月 1 日。

函数 isLeapYear(int y)的功能是判断 y 代表的年份是否为闰年，是则返回 1，否则返回 0。

DATE 类型定义如下:

```
typedef  struct  {
        int  year ，month ，day;
}DATE;
```

【C 函数 1】

```
int  GetDateld(  DATE  date  )
{
```

```
    const  int  days _month[13]  =  {  0，31，28，31，30，31，30，31，31，30，
    31，30 ，31 );
       int i ，date_id = date.day;
       for (  i =  0;  i <   (1) ; i++ )
             date_id  +=   days_month[i];
       if (  (2)  && isLeapYear(date.year)   )  date_id++;
       return date_id;
  }
```

【C 函数 2】

```
    __(3)__Kday_Date(int   theyear ，int  k)
  {
  int i;
  DATE  date;
     int  days_month(13) = { 0，31，28，31，30，31，30，31，31，30，31，30，31};
     assert(k>=l  &&  k<=365  &&  theyear>=1900);   /*不满足断言时程序终止*/
     date .year =   (4)    ;
     if   (isLeapYear(date.year))     days_month[2]++;
     for (i=1;  ;  )   {
        k = k - days_month(i++);
        if  (k<=0) { date .day =  k +   (5)   ; date.month  = i-1;  break; }
     }
     return date;
   }
```

● 阅读以下说明和 C 程序，填充程序中的空缺，将解答填入答题纸的对应栏内。

【说明】

埃拉托斯特尼筛法求不超过自然数 N 的所有素数的做法是:先把 N 个自然数按次序排列起来，1 不是素数，也不是合数，要划去; 2 是素数，取出 2(输出)，然后将 2 的倍数都划去:剩下的数中最小者为 3，3 是素数，取出 3(输出)，再把 3 的倍数都划去;剩下的数中最小者为 5，5 是素数，再把 5 的倍数都划去。这样一直做下去，就会把不超过 N 的全部合数都筛掉，每次从序列中取出的最小数所构成的序列就是不超过 N 的全部质数。

下面的程序实现埃拉托斯特尼筛法求素数，其中，数组元素 sieve[i](i>0)的下标 i 对应自然数 i，sieve[i]的值为 1/0 分别表示 i 在/不在序列中，也就是将 i 划去(去掉)时，就将 sieve[i]设置为 0。

【C 程序】

```
   *include <stdio.h>
   *define N 10000
   int main __(3)__
   {
      char sieve[N+1] = {0};
      int i =  0，k;
      /*初始时 2～N 都放入 sieve 数组*/
      for(i=2;   ___(1)___ ; i++)
            sieve[i] = 1;

      for( k  =  2; ;){
         /*找出剩下的数中最小者并用 k 表示*/
         for( ; k<N+1&& sieve[k]==0;   ____(2)____);
         if (   ___(3)___   )  break;
```

```
        printf("%d\t"，k);      /*输出素数*/
        /*从 sieve 中去掉 k 及其倍数*/
        for( i=k; i<N+1;  i=  (4)  ）
            (5)          ;
    }/*end of  for*/

     return  0;
}  /*end  of  main*/
```

● 阅读以下说明和 C 程序，填充函数中的空缺，将解答填入答题纸的对应栏内。

【说明】

N 个游戏者围成一圈，从 l～N 顺序编号，游戏方式如下:从第一个人开始报数(从 1 到 3 报数)，凡报到 3 的人退出圈子，直到剩余一个游戏者为止，该游戏者即为获胜者。

下面的函数 playing(LinkList head)模拟上述游戏过程并返回获胜者的编号。其中，N 个人围成的圈用一个包含 N 个结点的单循环链表来表示，如图 4-1 所示，游戏者的编号放在结点的数据域中。

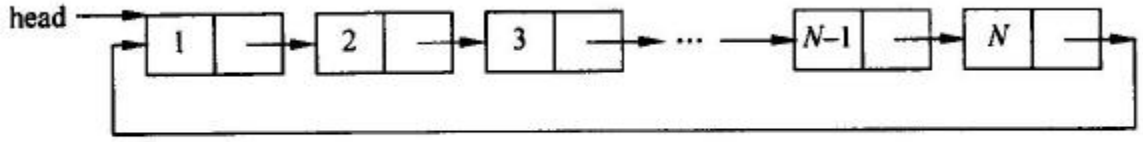


图 4-1

在函数中，以删除结点来模拟游戏者退出圈子的处理。整型变量 c(初值为 1)用于计数，指针变量 p 的初始值为 head(如图 4-1 所示)。游戏时，从 p 所指向的结点开始计数，p 沿链表中的指针方向遍历结点，c 的值随 p 的移动相应地递增。当 c 计数到 2 时，就删除 p 所指结点的下一个结点(因下一个结点就表示报数到 3 的游戏者)，如图 4-2 所示，然后将 c 设置为 0 后继续游戏过程。

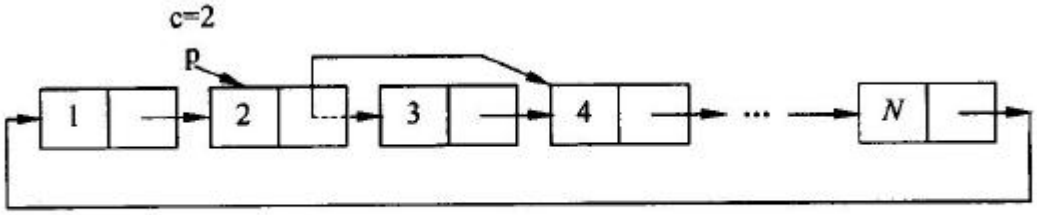


图 4-2

结点类型定义如下:

```
typedef  struct node{
   int  code;      /*游戏者的编号*/
   struct  node  *next;
}NODE，*LinkList;
```

【C 函数】

```
   int  playing(LinkList  head ，int  n)
   {   /* head 指向含有 n 个结点的循环单链表的第一个结点(即编号为 1 的游戏者) */
   LinkList  p = head ，q;
```

```
    int theWinner ， c = 1;

 whi1e  ( n >  (1) ){
       if (c == 2) { /*当 c 等于 2 时，p 所指向结点的后继即为将被删除的结点*/
          q  =  p->next;
          p->next  =   (2)   ;
          printf("%d\t" ，q->code);    /*输出退出圈子的游戏者编号时*/
          free (q) ;
          c =    (3)    ;
           n--;
        } /*if*/
        p =  (4)      ;
        c++;
     }/*while*/
     theWinner=  （5）  ；
free（p）；
    return   theWinner; /*返回最后一个游戏者（即获胜者）的编号*/
}
```

● 阅读以下说明和 Java 程序，填充代码中的空缺，将解答填入答题纸的对应栏内。

【说明】

某学校在学生毕业时要对其成绩进行综合评定，学生的综合成绩(*GPA*)由其课程加权平均成绩(*Wg*)与附加分(*Ag*)构成，即 *GPA= Wg +Ag*。

设一个学生共修了 n 门课程，则其加权平均成绩 (*Wg*) 定义如下:

$$Wg = \frac{\sum_{i=1}^{n} grade_i \times C_i}{\sum_{i=1}^{n} C_i}$$

其中，$grade_i$、C_i 分别表示该学生第 i 门课程的百分制成绩及学分。

学生可以通过参加社会活动或学科竞赛获得附加分(*Ag*)。学生参加社会活动所得的活动分(Apoints)是直接给出的，而竞赛分(*Awards*)则由下式计算(一个学生最多可参加 m 项学科竞赛):

$$Awards = \sum_{i=1}^{m} l_i \times s_i$$

其中，l_i 和 S_i 分别表示学生所参加学科竞赛的级别和成绩。

对于社会活动和学科竞赛都不参加的学生，其附加分按活动分为 0 计算。

下面的程序实现计算学生综合成绩的功能，每个学生的基本信息由抽象类 Student 描述，包括学号(stuNo)、姓名(name) 、课程成绩学分(grades) 和综合成绩 (GPA)等，参加社会活动的学生由类 ActStudent 描述，其活动分由 Apoints 表示，参加学科竞赛的学生由类 CmpStudent 描述，其各项竞赛的成绩信息由 awards 表示。

【 C++代码】

```
    #include  <string>
    #include  <iostream>
    using namespace std;
    const  int  N=5;      /*课程数*/
```

```
const int M=2;      /*竞赛项目数*/

class Student{
protected:
    int  stuNo; string name;
double GPA;          /*综合成绩*/
int (*grades) [2];    /*各门课程成绩和学分*/
public:
    Student ( const int stuNo，const string &name，int grades[ ] [2] ){
        this->stuNo = stuNo; grades; this->name = name; this->grades =
        grades；
    }
virtual ~Student( ) { }
    int getStuNo( )   {/*实现略*/
string getName( )     {/*实现略*/ }
     (1)   ;
    double computeWg ( ){
        int totalGrades = 0，totalCredits = 0;
        for (int i = 0; i < N; i++) {
            totalGrades += grades [i] [0] * grades [i] [1];  totalCredits +=
            grades [i] [1];
        }
        return GPA =(double)totalGrades / totalCredits;
    }
};
class ActStudent : public Student {
int Apoints;
public:
    ActStudent(const int stuNo， const string &name， int gs[ ] [2]， int
    Apoints)
:  (2) {
       this-＞Apoints = Apoints；
    }
    double getGPA ( ) { return GPA = (3) ;}
};
class CmpStudent: public Student{
private:
int (*awards) [2];
public:
  CmpStudent (const int stuNo，const string &name，int gs[] [2]，int
  awards [ ] [2])
  : (4)  {this->awards = awards;}
  double getGPA( )  {
        int Awards = 0;
        for (int i = 0; i < M;  i++)  {
Awards += awards [i] [0]  * awards [i] [1];
}
        return GPA =  (5) ;
   }
}
int main ( )
```

```
{  //以计算 3 个学生的综合成绩为例进行测试
int g1[ ] [2]  =  {{80 ，  3} ，  {90 ，  2} ，  {95 ，  3} ，  {85 ，  4} ，  {86 ，  3}} ，
    g2[ ][2]  =  {{60 ，  3} ，  {60 ，  2} ，  {60 ，  3} ，  {60，  4} ，  {65，  3}} ，
      g3[ ] [2] = {{80，  3} ，  {90，  2} ，  (70 ，  3} ，  {65，  4} ，  {75，  3}}; //课程成绩
   int  c3[ ] [2] = {{2 ，  3} ，  {3，  3}};                                    //竞赛成绩
   Student*  students[3]  =  {
      new  ActStudent  (101，  "John" ，  g1，  3) ，  //3 为活动分
      new  ActStudent  (102 ，  "Zhang" ，  g2 ，  0),
      new  CmpStudent  (103 ，  "Li" ，  g3 ，  c3) ，
   };
//输出每个学生的综合成绩
   for(int  i=0;  i<3;  i++)
         cout<<  (6)   <<endl;
   delete  *students;
   return 0;
```

● 阅读以下说明和 Java 程序，填充代码中的空缺，将解答填入答题纸的对应栏内。

【说明】

某学校在学生毕业时要对其成绩进行综合评定，学生的综合成绩(*GPA*)由其课程加权平均成绩(*Wg*)与附加分(*Ag*)构成，即 *GPA= Wg +Ag*。

设一个学生共修了 n 门课程，则其加权平均成绩 (*Wg*) 定义如下：

$$Wg = \frac{\sum_{i=1}^{n} grade_i \times C_i}{\sum_{i=1}^{n} C_i}$$

其中，$grade_i$、C_i 分别表示该学生第 i 门课程的百分制成绩及学分。

学生可以通过参加社会活动或学科竞赛获得附加分(*Ag*)。学生参加社会活动所得的活动分(Apoints)是直接给出的，而竞赛分(*Awards*)则由下式计算(一个学生最多可参加 m 项学科竞赛)：

$$Awards = \sum_{i=1}^{m} l_i \times s_i$$

其中，l_i 和 S_i 分别表示学生所参加学科竞赛的级别和成绩。

对于社会活动和学科竞赛都不参加的学生，其附加分按活动分为 0 计算。

下面的程序实现计算学生综合成绩的功能，每个学生的基本信息由抽象类 Student 描述，包括学号(stuNo)、姓名(name) 、课程成绩学分(grades) 和综合成绩 (GPA)等，参加社会活动的学生由类 ActStudent 描述，其活动分由 Apoints 表示，参加学科竞赛的学生由类 CmpStudent 描述，其各项竞赛的成绩信息由 awards 表示。

【Java 代码】

```
abstract  class  Student  {
      protected  String  name;
      protected  int  stuNo;
      protected double GPA;                  /*综合成绩*/
```

```
        protected int [ ] [ ] grades;          /*各门课程成绩和学分*/
    //其他信息略
    public Student ( int stuNo ， String name ， int[ ] [ ] grades) {
        this. stuNo = stuNo; grades;    this.name = name;    this.grades =
        grades;
    }
     ___(1)___ ;
    double computeWg( ){
        int totalGrades = 0 ， totalCredits = 0;
        for (int i = 0; i < grades.length; i++) {
            totalGrades += grades [i] [0] * grades [i] [1] ;
            totalCredits += grades [i] [1];
        }
        return (double)totalGrades / totalCredits;
    }
}

    class ActStudent extends Student {
        private int Apoints;
        ActStudent (int stuNo ， String name ， int[] [] grades ， int Apoints){
            _____(2)____;
            this. Apoints = Apoints;
        }
        public double getGPA( ) { return GPA =___(3)___;
}
}

class CmpStudent extends Student {
    private int [ ] [ ] Awards;
    CmpStudent (int stuNo ， String name ， int[ ] [ ] grades ， int[ ] [ ] awards) {
        ___(4)__;
        this.Awards = awards;
    }
    public double getGPA__(6)__ {
        int totalAwards = 0;
        for (int i = 0; i < awards.length; i++) {
totalAwards += awards[i] [0] * awards[i] [1];
}
        return GPA =__(5)___;
    }
}
    public class GPASystem ( //以计算 3 个学生的综合成绩为倒进行测试
    public static void main(String[] args) {
        int [ ] [ ] gl = {{80，3} ，{90，2} ，{95，3} ，{85，4} ，(86 ，3}) ，
g2 = {{60，3} ，{60，2} ，{60，3} ，{60，4} ，(65 ，3}) ，
    g3 = {{80，3} ，{90，2} ，{70，3} ，{65，4} ，(75 ，3}); //课程成绩
        int [ ][ ] e1 = ((2 ，3) ，{1，2}} ，e2 = {{1，3}}; //竞赛成绩
          Student students[ ] = {
new ActStudent (101，"John" ，gl ，3)， / /3 为活动分
              new ActStudent (102 ，"Zhang" ，g2 ，0) ，
              new CmpStudent (103，"Li"，g3，e2)};
```

```
        }
    //输出每个学生的综合成绩
        for (int i = 0; i < students .length; i++) (
            System.out,println(___(6)____);
        }
}
}
```