初赛模赛

选择题

选择

1 如果128种颜色用二进制编码来表示，至少需要 (　) 位。

A. 5　　B. 6　　C. 7　　D. 8

答案：C

解析：每一位可以表示0或1，从0000000到1111111，7位一共可以表示2^7=128种。

选择

2 若12*25=311成立，则用的是（　）进制。

A. 11　　B. 8　　C. 7　　D.9

答案：D

解析：(1*n+2)* (2*n+5) = $3n^2$+1*n+1。n^2-8n-9=0，解得n = 9或n = -1。

选择

3 一个袋子里装了100个苹果，100个香蕉，100个橘子，100个梨子。从袋子中取出一个水果需要1分钟，那么需要（　）分钟就能肯定至少已经拿出1打（12个）相同种类的水果。

A. 12　　B. 13　　C. 45　　D. 101

答案：C

解析：根据鸽巢原理，4*11+1=45。

选择

4 下列程序段的时间复杂度是（　）。

```
int cnt=0;
for(int i=1; i<n; i*=2)
        for(int j=1; j<n; j++)
                cnt++;
```

A. $O(\log_2(n))$　　B. $O(n)$　　C. $O(n*\log_2(n))$　　D.$O(n^2)$

答案：C

解析：时间复杂度计算。对外层的for循环，设其执行次数为t，则$2^t<n$，即$t<\log_2(n)$。内层for循环需要执行n次，于是整段程序的时间复杂度是$O(n*\log_2(n))$。

选择

5 以下是32位机器与64位机器区别的是（　）。

A. 硬盘大小不同
B. 显示器分辨率不同
C. 操作系统版本号不同
D. 寻址空间不同

答案：D

解析：32位机器与64位机器的区别是寻址空间不同。

选择

6 如果一棵二叉树的先序遍历是ACDBEFG，中序遍历是DCAEBFG，那么它的后序遍历是（ ）。

A. DCEGFBA
B. DCAEFGB
C. DCEFGBA
D. DCAEGFB

答案：A

解析：先序：根-左-右。中序：左-根-右。先序中首个结点即为根（在本题中为A），然后通过中序里根A的所在位置，划分左右子树（根分左右）；再根据左右子树的**结点个数**，去划分得到先序的左右子树。递归进行上述步骤。可画出树的形态，最后再求后序遍历。

选择

7如果开始时计算机处于小写输入状态，现在有一只小狗反复按照 CapsLock、字母键 A、字母键 B 、字母键 C 、字母键 D的顺序来回按键，即 CapsLock、A、B、C、D、CapsLock、A、B、C、D、CapsLock、A、B、C、D、……，屏幕上**输出的**第 2020 个字符是字母（　）。

A. 大写A　　B. 大写B　　C. 大写C　　D. 大写D

答案：D

解析：CapsLock是大小写切换。所以，每组一共八个字母，即ABCDabcd。2020%8=4，第2020个输出的是大写D

选择

8 如果进栈序列e1,e2,e3,e4，则不可能的出栈序列是：（ ）。

A. e2, e4, e3, e1
B. e4, e3, e2, e1
C. e1, e2, e3, e4
D. e3, e1, e4, e2

答案：D

解析：e3 先出栈，则e1 不可能在e2 之前出栈，D 错误。

选择

9 有如下程序:

```
#include <iostream>
using namespace std;
int a[3][3];
int main(){
    for(int i=0;i<3;i++)
        for(int j=0;j<3;j++) a[i][j]=i*3+j+1;
    for(int i=1;i<3;i++)
        for(int j=0;j<2;j++) cout<< a[j][i];
     return 0;
}
```

运行后的输出结果是（　）。

A. 2536　　B. 2356　　C. 4758　　D. 4578

答案：A

解析：注意输出的是a[j][i]，而不是a[i][j]。所以依次输出的是a[0][1]、a[1][1]、a[0][2]、a[1][2]。

选择

10 8名男生和4名女生围绕圆桌就坐,任意两个女生不相邻的坐法有（ ）种。

A. 67737600　　B. 8467200
C. 40320　　D. 352800

答案：B

解析：男生圆排列，8! / 8 = 7!，固定一个男生，把女生插空到8名男生之间：A(8,4)。
方案数7! * A(8,4)=8467200。

选择

11 关于下面说法，错误的一项是（ ）。

A. ASCII码是一种字符编码,常用7位码
B. 程序设计的三种基本结构是顺序、选择、循环
C. 刷新率、CPI、DPI均可以用于描述鼠标性能
D. 目前世界上最大的计算机互联网络是IBM网

答案：D

解析：Internet是目前世界上覆盖范围最广、使用者最多、最成功的计算机互联网络。

选择

12 小红课桌上有5本书，每本书都放有一个写着书名的书签。调皮的小王把小红的书签全部打乱了。那么，每个书签都没有插到对应的书中的可能性总共有（ ）种。

A. 38　　B. 40　　C. 44　　D. 48

答案：C

解析：经典的错排问题，假设n本书和书签全打乱的可能性一共是f(n)种，那么遵循如下递推公式
f(n) = (n-1) * (f(n-1)+f(n-2))，所以f(5)=44。

选择

13设 G 是有 7 个结点的完全图，要得到一棵生成树，需要从 G 中删去（　）条边。

A. 10　　B. 12　　C. 15　　D. 18

答案：C

解析：7个点的完全图有7*6/2=21条边，而7个点的树有7-1=6条边，因此需要删去21-6=15条边。

选择

14 46! 的计算结果，尾数总共有（　）个零。

A. 9　　B. 10　　C. 11　　D.12

答案：B

解析：求46! 尾数总共有几个零，也就是求46! 中2*5的因子的个数。由于因子2的数量远多于5的个数，所以就转换成求5的因子个数，1到46中有5、10、15、20、25（5*5）、30、35、40、45，总计10个。

选择

15定义语句“double * array[8]”的含义正确的是（　）。

A. array是一个指针，它指向一个数组，数组的元素是双精度浮点型。
B. array是一个数组，数组的每一个元素是指向双精度浮点型数据的指针
C. C++语言中不允许这样的定义语句
D. 以上都不对

答案：B

解析：根据数组定义，array是一个数组，包含了8个double型指针。

阅读程序

2分钟

阅读程序

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int max, min, sum, cnt = 0;
5      int temp;
6      cin >> temp;
7      if (temp == 0) return 0;
8      max = min = sum = temp;
9      cnt++;
10     while (temp != 0) {
11         cin >> temp;
12         if (temp != 0) {
13             sum += temp;
14             cnt++;
15             if (temp > max) max = temp;
16             if (temp < min) min = temp;
17         }
18     }
19     cout<<cnt<<" "<<min<<" "<<max<<" "<<sum/cnt<<endl;
20     return 0;
21 }
```

1	识别变量
常见变量名 翻译循环变量 根据变量名的英文推断	

2	找出关键语句
控制结构(for, if) 常见算法的基本操作 函数参数、返回值	

3	理解代码段作用
翻译解释代码段	

阅读程序

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int max, min, sum, cnt = 0;
5      int temp;
6      cin >> temp;
7      if (temp == 0) return 0;
8      max = min = sum = temp;
9      cnt++;
10     while (temp != 0) {
11         cin >> temp;
12         if (temp != 0) {
13             sum += temp;
14             cnt++;
15             if (temp > max) max = temp;
16             if (temp < min) min = temp;
17         }
18     }
19     cout<<cnt<<" "<<min<<" "<<max<<" "<<sum/cnt<<endl;
20     return 0;
21 }
```

解释变量的作用

max	最大值
min	最小值
sum	各数总和
cnt	数字个数
temp	临时变量

阅读程序

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int max, min, sum, cnt = 0;
5      int temp;
6      cin >> temp;
7      if (temp == 0) return 0;
8      max = min = sum = temp;
9      cnt++;
10     while (temp != 0) {
11         cin >> temp;
12         if (temp != 0) {
13             sum += temp;
14             cnt++;
15             if (temp > max) max = temp;
16             if (temp < min) min = temp;
17         }
18     }
19     cout<<cnt<<" "<<min<<" "<<max<<" "<<sum/cnt<<endl;
20     return 0;
21 }
```

关键语句

定义变量

首个数字输入到temp

如果数字为0直接结束

while循环
逐个读入数字到temp
并更新最大值、最小值、
数字个数、总和

注意关键条件

遇到0结束while

依次输出个数、最小值、
最大值、平均值

阅读程序

```
#include <iostream>
using namespace std;
int main() {
    int max, min, sum, cnt = 0;
    int temp;
    cin >> temp;
    if (temp == 0) return 0;
    max = min = sum = temp;
    cnt++;
    while (temp != 0) {
        cin >> temp;
        if (temp != 0) {
            sum += temp;
            cnt++;
            if (temp > max) max = temp;
            if (temp < min) min = temp;
        }
    }
    cout<<cnt<<" "<<min<<" "<<max<<" "<<sum/cnt<<endl;
    return 0;
}
```

判断	1. 本程序统计了输入数字个数、最大值、最小值、平均值，输入遇-1结束。(　)

阅读程序

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int max, min, sum, cnt = 0;
5      int temp;
6      cin >> temp;
7      if (temp == 0) return 0;
8      max = min = sum = temp;
9      cnt++;
10     while (temp != 0) {
11         cin >> temp;
12         if (temp != 0) {
13             sum += temp;
14             cnt++;
15             if (temp > max) max = temp;
16             if (temp < min) min = temp;
17         }
18     }
19     cout<<cnt<<" "<<min<<" "<<max<<" "<<sum/cnt<<endl;
20     return 0;
21 }
```

判断	2. 对任意的整数值x，只需要将第7行中if条件判断修改为temp==x，就可以将本程序功能变成为输入遇值x结束。（　）

阅读程序

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int max, min, sum, cnt = 0;
5      int temp;
6      cin >> temp;
7      if (temp == 0) return 0;
8      max = min = sum = temp;
9      cnt++;
10     while (temp != 0) {
11         cin >> temp;
12         if (temp != 0) {
13             sum += temp;
14             cnt++;
15             if (temp > max) max = temp;
16             if (temp < min) min = temp;
17         }
18     }
19     cout<<cnt<<" "<<min<<" "<<max<<" "<<sum/cnt<<endl;
20     return 0;
21 }
```

判断	3. 去掉第8行的连续赋值语句，或者忘记对第4行的整型变量cnt初始化为0，这两种情况都可能影响程序最终输出结果。(　　)

阅读程序

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int max, min, sum, cnt = 0;
5      int temp;
6      cin >> temp;
7      if (temp == 0) return 0;
8      max = min = sum = temp;
9      cnt++;
10     while (temp != 0) {
11         cin >> temp;
12         if (temp != 0) {
13             sum += temp;
14             cnt++;
15             if (temp > max) max = temp;
16             if (temp < min) min = temp;
17         }
18     }
19     cout<<cnt<<" "<<min<<" "<<max<<" "<<sum/cnt<<endl;
20     return 0;
21 }
```

判断	4. 将15行if条件判断语句改成temp>=max，并且把16行if条件判断语句改成temp<=min，对程序结果没有任何影响。(　　)

阅读程序

```
#include <iostream>
using namespace std;
int main() {
    int max, min, sum, cnt = 0;
    int temp;
    cin >> temp;
    if (temp == 0) return 0;
    max = min = sum = temp;
    cnt++;
    while (temp != 0) {
        cin >> temp;
        if (temp != 0) {
            sum += temp;
            cnt++;
            if (temp > max) max = temp;
            if (temp < min) min = temp;
        }
    }
    cout<<cnt<<" "<<min<<" "<<max<<" "<<sum/cnt<<endl;
    return 0;
}
```

选择	5. 若输入0 1 2 3 4 5，程序运行到第 (　　) 行结束。
	A. 6　B. 7　C. 10　D. 20

阅读程序

```
#include <iostream>
using namespace std;
int main() {
    int max, min, sum, cnt = 0;
    int temp;
    cin >> temp;
    if (temp == 0) return 0;
    max = min = sum = temp;
    cnt++;
    while (temp != 0) {
        cin >> temp;
        if (temp != 0) {
            sum += temp;
            cnt++;
            if (temp > max) max = temp;
            if (temp < min) min = temp;
        }
    }
    cout<<cnt<<" "<<min<<" "<<max<<" "<<sum/cnt<<endl;
    return 0;
}
```

选择 6. 若输入1 2 3 4 6 0 8 9，程序最终输出结果是（ ）。

A. 5 1 6 3.2　　B. 8 1 9 4　　C. 6 1 6 3　　D. 5 1 6 3

阅读程序

3分钟

阅读程序

```
6   #include <stdio.h>
7   int gcd(int a, int b){
8       if(b == 0) return a;
9       return gcd(b,a%b);
10  }
11  void work(int a, int b) {
12      int s[10009], t[10009], i = 0, d = 1, j;
13      while(1) {
14          if(a == 0) break;
15          a *= 10; //除法借位
16          t[i] = a;
17          s[i] = a / b;
18          a %= b;
19          for(j = 0; j < i; ++j)
20              if(t[j] == t[i]) { d = 0; break; }
21          if(d == 0) break;
22          printf("%d", s[i]); i++;
23      }
24  }
25  int main() {
26      int a, b, g; scanf("%d%d", &a, &b);
27      if(a > b)  g = gcd(a, b);
28      else  g = gcd(b, a);
29      a /= g; b /= g; printf("%d.", a/b);
30      a %= b; work(a, b);
31      return 0;
```

1	识别变量
常见变量名 翻译循环变量 根据变量名的英文推断	

2	找出关键语句
控制结构(for, if) 常见算法的基本操作 函数参数、返回值	

3	理解代码段作用
翻译解释代码段	

阅读程序

```
6   #include <stdio.h>
7   int gcd(int a, int b){
8       if(b == 0) return a;
9       return gcd(b,a%b);
10  }
11  void work(int a, int b) {
12      int s[10009], t[10009], i = 0, d = 1, j;
13      while(1) {
14          if(a == 0) break;
15          a *= 10; //除法借位
16          t[i] = a;
17          s[i] = a / b;
18          a %= b;
19          for(j = 0; j < i; ++j)
20              if(t[j] == t[i]) { d = 0; break; }
21          if(d == 0) break;
22          printf("%d", s[i]); i++;
23      }
24  }
25  int main() {
26      int a, b, g; scanf("%d%d", &a, &b);
27      if(a > b)  g = gcd(a, b);
28      else  g = gcd(b, a);
29      a /= g; b /= g; printf("%d.", a/b);
30      a %= b; work(a, b);
31      return 0;
```

解释变量的作用

a	分子
b	分母
i	计数器变量
t[i]	第i位上10倍a的值
s[i]	求第i位上10*a除以b的值 得到第i小数位上真实值
d	标记是否已经到了第一个 **重复值**所在位置

本程序输入一个分数的
正整数分子a和正整数分母b
将其转换为小数形式输出

阅读程序

```
6   #include <stdio.h>
7   int gcd(int a, int b){
8       if(b == 0) return a;
9       return gcd(b,a%b);
10  }
11  void work(int a, int b) {
12      int s[10009], t[10009], i = 0, d = 1, j;
13      while(1) {
14          if(a == 0) break;
15          a *= 10; //除法借位
16          t[i] = a;
17          s[i] = a / b;
18          a %= b;
19          for(j = 0; j < i; ++j)
20              if(t[j] == t[i]) { d = 0; break; }
21          if(d == 0) break;
22          printf("%d", s[i]); i++;
23      }
24  }
25  int main() {
26      int a, b, g; scanf("%d%d", &a, &b);
27      if(a > b)  g = gcd(a, b);
28      else  g = gcd(b, a);
29      a /= g; b /= g; printf("%d.", a/b);
30      a %= b; work(a, b);
31      return 0;
```

关键语句

递归返回a和b的最大公约数

定义并输入分子a和分母b

调用gcd函数求a和b最大公约数

将a和b约分到互质
先输出a/b整数部分和小数点

真分数部分调用work求小数表示

阅读程序

```
6   #include <stdio.h>
7   int gcd(int a, int b){
8       if(b == 0) return a;
9       return gcd(b,a%b);
10  }
11  void work(int a, int b) {
12      int s[10009], t[10009], i = 0, d = 1, j;
13      while(1) {
14          if(a == 0) break;
15          a *= 10; //除法借位
16          t[i] = a;
17          s[i] = a / b;
18          a %= b;
19          for(j = 0; j < i; ++j)
20              if(t[j] == t[i]) { d = 0; break; }
21          if(d == 0) break;
22          printf("%d", s[i]); i++;
23      }
24  }
25  int main() {
26      int a, b, g; scanf("%d%d", &a, &b);
27      if(a > b)  g = gcd(a, b);
28      else  g = gcd(b, a);
29      a /= g; b /= g; printf("%d.", a/b);
30      a %= b; work(a, b);
31      return 0;
32  }
```

关键语句

递归返回a和b的最大公约数

定义数组s,t和变量i,d,j

while循环逐位计算小数表示

a==0表示已经除尽，结束循环

除法借位乘10，存到t[i]中

计算对应位小数值，存到s[i]中

检查i位a值是否已在前序位出现
若出现则到**重复值**，标记并结束

否则继续输出第i位小数值s[i]

定义并输入分子a和分母b

调用gcd函数求a和b最大公约数

将a和b约分到互质
先输出a/b整数部分和小数点

真分数部分调用work求小数表示

阅读程序

```
6  #include <stdio.h>
7  int gcd(int a, int b){
8      if(b == 0) return a;
9      return gcd(b,a%b);
10 }
11 void work(int a, int b) {
12     int s[10009], t[10009], i = 0, d = 1, j;
13     while(1) {
14         if(a == 0) break;
15         a *= 10; //除法借位
16         t[i] = a;
17         s[i] = a / b;
18         a %= b;
19         for(j = 0; j < i; ++j)
20             if(t[j] == t[i]) { d = 0; break; }
21         if(d == 0) break;
22         printf("%d", s[i]); i++;
23     }
24 }
25 int main() {
26     int a, b, g; scanf("%d%d", &a, &b);
27     if(a > b)  g = gcd(a, b);
28     else  g = gcd(b, a);
29     a /= g; b /= g; printf("%d.", a/b);
30     a %= b; work(a, b);
31     return 0;
32 }
```

判断	1. 函数gcd使用了递归的方法，返回参数a和b的最小公倍数。(　　)

阅读程序

```
6  #include <stdio.h>
7  int gcd(int a, int b){
8      if(b == 0) return a;
9      return gcd(b,a%b);
10 }
11 void work(int a, int b) {
12     int s[10009], t[10009], i = 0, d = 1, j;
13     while(1) {
14         if(a == 0) break;
15         a *= 10; //除法借位
16         t[i] = a;
17         s[i] = a / b;
18         a %= b;
19         for(j = 0; j < i; ++j)
20             if(t[j] == t[i]) { d = 0; break; }
21         if(d == 0) break;
22         printf("%d", s[i]); i++;
23     }
24 }
25 int main() {
26     int a, b, g; scanf("%d%d", &a, &b);
27     if(a > b)  g = gcd(a, b);
28     else  g = gcd(b, a);
29     a /= g; b /= g; printf("%d.", a/b);
30     a %= b; work(a, b);
31     return 0;
32 }
```

判断　2. 30行作为work函数的参数传入a和b，一定满足a<b且a≠0。(　　)

阅读程序

```
6  #include <stdio.h>
7  int gcd(int a, int b){
8      if(b == 0) return a;
9      return gcd(b,a%b);
10 }
11 void work(int a, int b) {
12     int s[10009], t[10009], i = 0, d = 1, j;
13     while(1) {
14         if(a == 0) break;
15         a *= 10; //除法借位
16         t[i] = a;
17         s[i] = a / b;
18         a %= b;
19         for(j = 0; j < i; ++j)
20             if(t[j] == t[i]) { d = 0; break; }
21         if(d == 0) break;
22         printf("%d", s[i]); i++;
23     }
24 }
25 int main() {
26     int a, b, g; scanf("%d%d", &a, &b);
27     if(a > b)  g = gcd(a, b);
28     else  g = gcd(b, a);
29     a /= g; b /= g; printf("%d.", a/b);
30     a %= b; work(a, b);
31     return 0;
32 }
```

判断	3. 去掉19-21行的代码，将增加程序发生运行时错误的风险。(　　)

阅读程序

```
6  #include <stdio.h>
7  int gcd(int a, int b){
8      if(b == 0) return a;
9      return gcd(b,a%b);
10 }
11 void work(int a, int b) {
12     int s[10009], t[10009], i = 0, d = 1, j;
13     while(1) {
14         if(a == 0) break;
15         a *= 10; //除法借位
16         t[i] = a;
17         s[i] = a / b;
18         a %= b;
19         for(j = 0; j < i; ++j)
20             if(t[j] == t[i]) { d = 0; break; }
21         if(d == 0) break;
22         printf("%d", s[i]); i++;
23     }
24 }
25 int main() {
26     int a, b, g; scanf("%d%d", &a, &b);
27     if(a > b)  g = gcd(a, b);
28     else  g = gcd(b, a);
29     a /= g; b /= g; printf("%d.", a/b);
30     a %= b; work(a, b);
31     return 0;
32 }
```

判断	4. 去掉27行、28行以及29行前两个语句，程序输出结果一定发生改变。(　　)

阅读程序

```
6  #include <stdio.h>
7  int gcd(int a, int b){
8      if(b == 0) return a;
9      return gcd(b,a%b);
10 }
11 void work(int a, int b) {
12     int s[10009], t[10009], i = 0, d = 1, j;
13     while(1) {
14         if(a == 0) break;
15         a *= 10; //除法借位
16         t[i] = a;
17         s[i] = a / b;
18         a %= b;
19         for(j = 0; j < i; ++j)
20             if(t[j] == t[i]) { d = 0; break; }
21         if(d == 0) break;
22         printf("%d", s[i]); i++;
23     }
24 }
25 int main() {
26     int a, b, g; scanf("%d%d", &a, &b);
27     if(a > b)  g = gcd(a, b);
28     else  g = gcd(b, a);
29     a /= g; b /= g; printf("%d.", a/b);
30     a %= b; work(a, b);
31     return 0;
32 }
```

选择	5.若输入91 13，输出结果是(　　)。
	A. 7　B. 7.　C. 7.0　D. 7.00

阅读程序

```
6  #include <stdio.h>
7  int gcd(int a, int b){
8      if(b == 0) return a;
9      return gcd(b,a%b);
10 }
11 void work(int a, int b) {
12     int s[10009], t[10009], i = 0, d = 1, j;
13     while(1) {
14         if(a == 0) break;
15         a *= 10; //除法借位
16         t[i] = a;
17         s[i] = a / b;
18         a %= b;
19         for(j = 0; j < i; ++j)
20             if(t[j] == t[i]) { d = 0; break; }
21         if(d == 0) break;
22         printf("%d", s[i]); i++;
23     }
24 }
25 int main() {
26     int a, b, g; scanf("%d%d", &a, &b);
27     if(a > b)  g = gcd(a, b);
28     else  g = gcd(b, a);
29     a /= g; b /= g; printf("%d.", a/b);
30     a %= b; work(a, b);
31     return 0;
32 }
```

选择 6. 以下哪个结果是程序可能输出的（ ）。

A. 10.0　　B. 1.571428571428　　C. 1.5714285　　D. 1.571428

阅读程序

3分钟

阅读程序

```
1  #include <iostream>
2  using namespace std;
3  int solve(int n, int m){
4      int i, sum;
5      if (m == 1) return 1;
6      sum = 0;
7      for (i = 1; i < n; i++)
8          sum += solve(i, m - 1);
9      return sum;
10 }
11 int main(){
12     int n, m;
13     cin>>n>>m;
14     cout<<solve(n, m)<<endl;
15     return 0;
16 }
```

1	识别变量
常见变量名 翻译循环变量 根据变量名的英文推断	

2	找出关键语句
控制结构(for, if) 常见算法的基本操作 函数参数、返回值	

3	理解代码段作用
翻译解释代码段	

阅读程序

```
1  #include <iostream>
2  using namespace std;
3  int solve(int n, int m){
4      int i, sum;
5      if (m == 1) return 1;
6      sum = 0;
7      for (i = 1; i < n; i++)
8          sum += solve(i, m - 1);
9      return sum;
10 }
11 int main(){
12     int n, m;
13     cin>>n>>m;
14     cout<<solve(n, m)<<endl;
15     return 0;
16 }
```

带入小数据

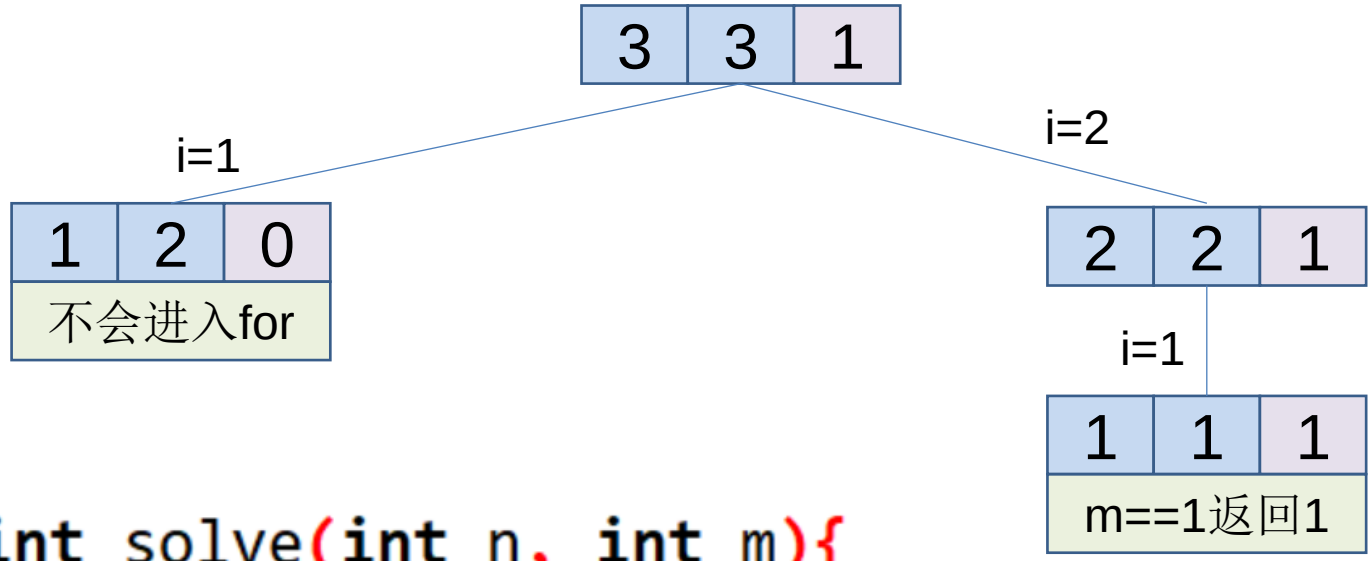
输入	1 1
输出	1

输入	1 2
输出	0

输入	2 2
输出	1

n=3, m=3

递归调用树

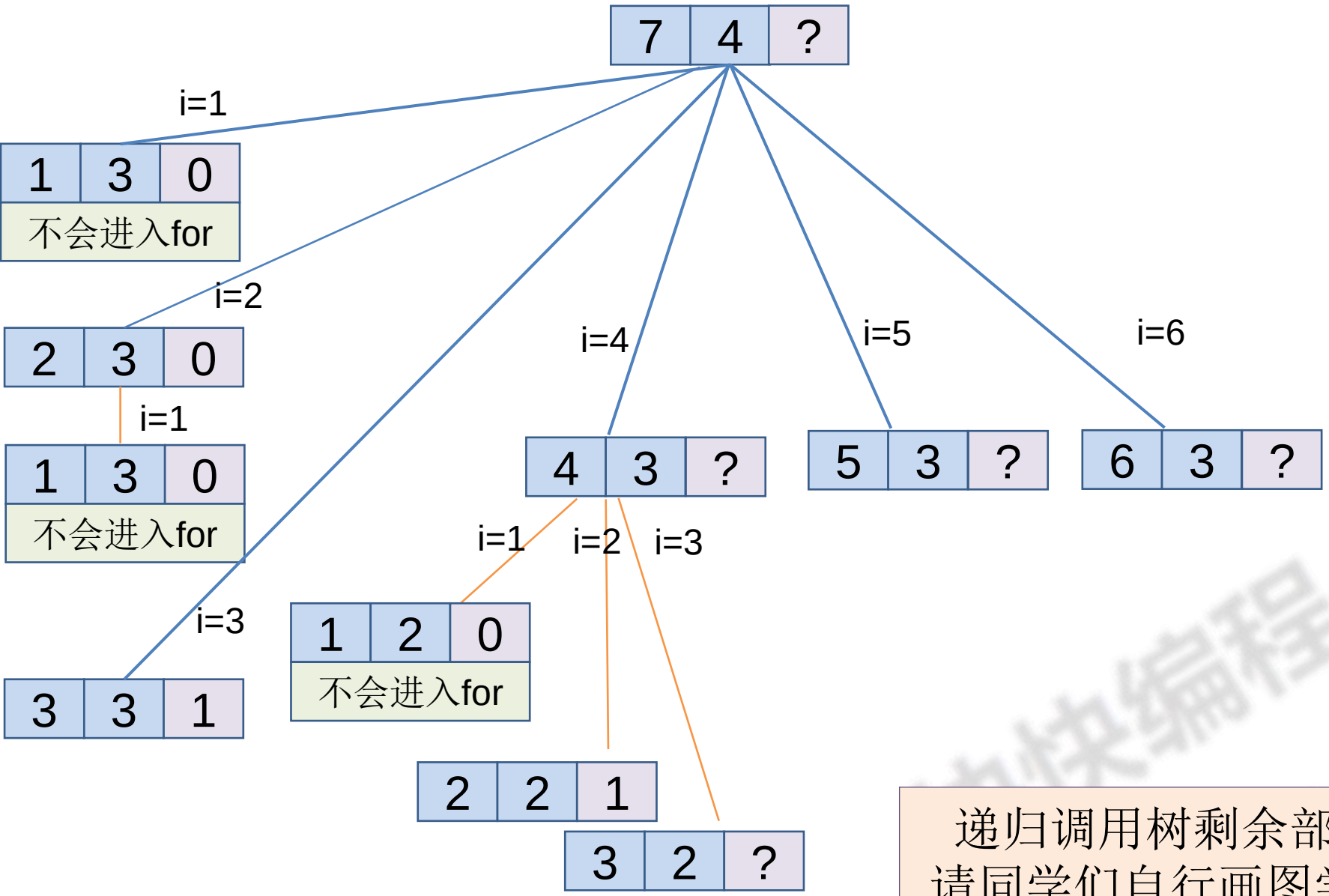



```
3  int solve(int n, int m){
4      int i, sum;
5      if (m == 1) return 1;
6      sum = 0;
7      for (i = 1; i < n; i++)
8          sum += solve(i, m - 1);
9      return sum;
10 }
```

n	m	return

n=7, m=4

递归调用树



递归调用树剩余部分
请同学们自行画图尝试

简化：使用表格递推

n\m	1	2	3	4
1		0	0	0
2			0	0
3				0
4				
5				
6				
7				

递归调用solve(n,m)
每深入一层m减1
n至少减1（for从1到n-1）
最终n减少到1无法进入for时
m必然还大于1，没有返回1
因此当n<m必然返回sum=0

```
3  int solve(int n, int m){
4      int i, sum;
5      if (m == 1) return 1;
6      sum = 0;
7      for (i = 1; i < n; i++)
8          sum += solve(i, m - 1);
9      return sum;
10 }
```

使用表格递推

n\m	1	2	3	4
1	1	0	0	0
2	1		0	0
3	1			0
4	1			
5	1			
6	1			
7	1			

当m==1时返回1

```
3  int solve(int n, int m){
4      int i, sum;
5      if (m == 1) return 1;
6      sum = 0;
7      for (i = 1; i < n; i++)
8          sum += solve(i, m - 1);
9      return sum;
10 }
```

使用表格递推

n\m	1	2	3	4
1	1	0	0	0
2	1	1	0	0
3	1			0
4	1			
5	1			
6	1			
7	1			

$s(n,m)=\Sigma\, s(i,m-1)$
即第n行m列的值等于第i行（i=1…n-1）第m-1的值之和

```
3  int solve(int n, int m){
4      int i, sum;
5      if (m == 1) return 1;
6      sum = 0;
7      for (i = 1; i < n; i++)
8          sum += solve(i, m - 1);
9      return sum;
10 }
```

使用表格递推

n\m	1	2	3	4
1	1	0	0	0
2	1	1	0	0
3	1	2		0
4	1			
5	1			
6	1			
7	1			

$s(n,m)=\Sigma\, s(i,m-1)$
即第n行m列的值等于第i行（**i=1…n-1**）第m-1的值之和

```
3  int solve(int n, int m){
4      int i, sum;
5      if (m == 1) return 1;
6      sum = 0;
7      for (i = 1; i < n; i++)
8          sum += solve(i, m - 1);
9      return sum;
10 }
```

使用表格递推

n\m	1	2	3	4
1	1	0	0	0
2	1	1	0	0
3	1	2		0
4	1	3		
5	1			
6	1			
7	1			

$s(n,m)=\Sigma\, s(i,m-1)$
即第n行m列的值等于第i行（**i=1…n-1**）第m-1的值之和

```
3  int solve(int n, int m){
4      int i, sum;
5      if (m == 1) return 1;
6      sum = 0;
7      for (i = 1; i < n; i++)
8          sum += solve(i, m - 1);
9      return sum;
10 }
```

使用表格递推

n\m	1	2	3	4
1	1	0	0	0
2	1	1	0	0
3	1	2	1	0
4	1	3	3	1
5	1	4	6	4
6	1	5	10	10
7	1	6	15	20

$s(n,m)=\Sigma\, s(i,m-1)$
即第n行m列的值等于第i行（**i=1…n-1**）第m-1的值之和

```
3  int solve(int n, int m){
4      int i, sum;
5      if (m == 1) return 1;
6      sum = 0;
7      for (i = 1; i < n; i++)
8          sum += solve(i, m - 1);
9      return sum;
10 }
```

使用表格递推

n\m	1	2	3	4
1	1	0	0	0
2	1	1	0	0
3	1	2	1	0
4	1	3	3	1
5	1	4	6	4
6	1	5	10	10
7	1	6	15	20

使用表格递推

n\m	1	2	3	4
1	1	0	0	0
2	1	1	0	0
3	1	2	1	0
4	1	3	3	1
5	1	4	6	4
6	1	5	10	10
7	1	6	15	20

总结递推规律！

当n>=2&&m>=2时

s(n,m)=s(n-1,m-1)+s(n-1,m)

阅读程序

```
1  #include <iostream>
2  using namespace std;
3  int solve(int n, int m){
4      int i, sum;
5      if (m == 1) return 1;
6      sum = 0;
7      for (i = 1; i < n; i++)
8          sum += solve(i, m - 1);
9      return sum;
10 }
11 int main(){
12     int n, m;
13     cin>>n>>m;
14     cout<<solve(n, m)<<endl;
15     return 0;
16 }
```

判断

1. 程序第三行改写成solve(int m, int n)，程序运行结果不变。(　　)

阅读程序

```
1  #include <iostream>
2  using namespace std;
3  int solve(int n, int m){
4      int i, sum;
5      if (m == 1) return 1;
6      sum = 0;
7      for (i = 1; i < n; i++)
8          sum += solve(i, m - 1);
9      return sum;
10 }
11 int main(){
12     int n, m;
13     cin>>n>>m;
14     cout<<solve(n, m)<<endl;
15     return 0;
16 }
```

判断

2. 去掉程序第6行后，程序运行结果不变。(　　)

阅读程序

```
1  #include <iostream>
2  using namespace std;
3  int solve(int n, int m){
4      int i, sum;
5      if (m == 1) return 1;
6      sum = 0;
7      for (i = 1; i < n; i++)
8          sum += solve(i, m - 1);
9      return sum;
10 }
11 int main(){
12     int n, m;
13     cin>>n>>m;
14     cout<<solve(n, m)<<endl;
15     return 0;
16 }
```

选择
3.若输入的n<m，则输出结果为 (　　)。
A. 0　　B. n C. m　　D. m+(m+1)+...+(n-1)

阅读程序

```
1  #include <iostream>
2  using namespace std;
3  int solve(int n, int m){
4      int i, sum;
5      if (m == 1) return 1;
6      sum = 0;
7      for (i = 1; i < n; i++)
8          sum += solve(i, m - 1);
9      return sum;
10 }
11 int main(){
12     int n, m;
13     cin>>n>>m;
14     cout<<solve(n, m)<<endl;
15     return 0;
16 }
```

选择
4. 输入4 3，程序第8行共循环（　）次。
A. 3　　B. 6　　C. 9　　D. 12

阅读程序

```cpp
#include <iostream>
using namespace std;
int solve(int n, int m){
    int i, sum;
    if (m == 1) return 1;
    sum = 0;
    for (i = 1; i < n; i++)
        sum += solve(i, m - 1);
    return sum;
}
int main(){
    int n, m;
    cin>>n>>m;
    cout<<solve(n, m)<<endl;
    return 0;
}
```

选择
5. 输入7 4，输出是（　）
A. 35　　B. 10　　C. 20　　D. 15

阅读程序

```cpp
#include <iostream>
using namespace std;
int solve(int n, int m){
    int i, sum;
    if (m == 1) return 1;
    sum = 0;
    for (i = 1; i < n; i++)
        sum += solve(i, m - 1);
    return sum;
}
int main(){
    int n, m;
    cin>>n>>m;
    cout<<solve(n, m)<<endl;
    return 0;
}
```

选择
6.（4分）以下四组输出数据中，输出最大的是（　）。
A. 6 3　　B. 5 3　　C. 7 5　　D. 8 7

完善程序

完善程序

（活动选择）学校在最近几天有n个活动，这些活动都需要使用学校的大礼堂，在同一时间，礼堂只能被一个活动使用。由于有些活动时间上有冲突，学校办公人员只好让一些活动放弃使用礼堂。现给出n个活动使用礼堂的起始时间begini和结束时间endi（begini<endi）。
程序在第一行输入一个整数n（n<=1000），接下来n行，每行两个整数，第1个是begini，第2个是endi（begini<endi<=32767），程序输出最多能安排的活动个数。

手算样例

输入样例：
3
1 4
3 5
4 5

输出多少？

输出样例：
2

手算样例

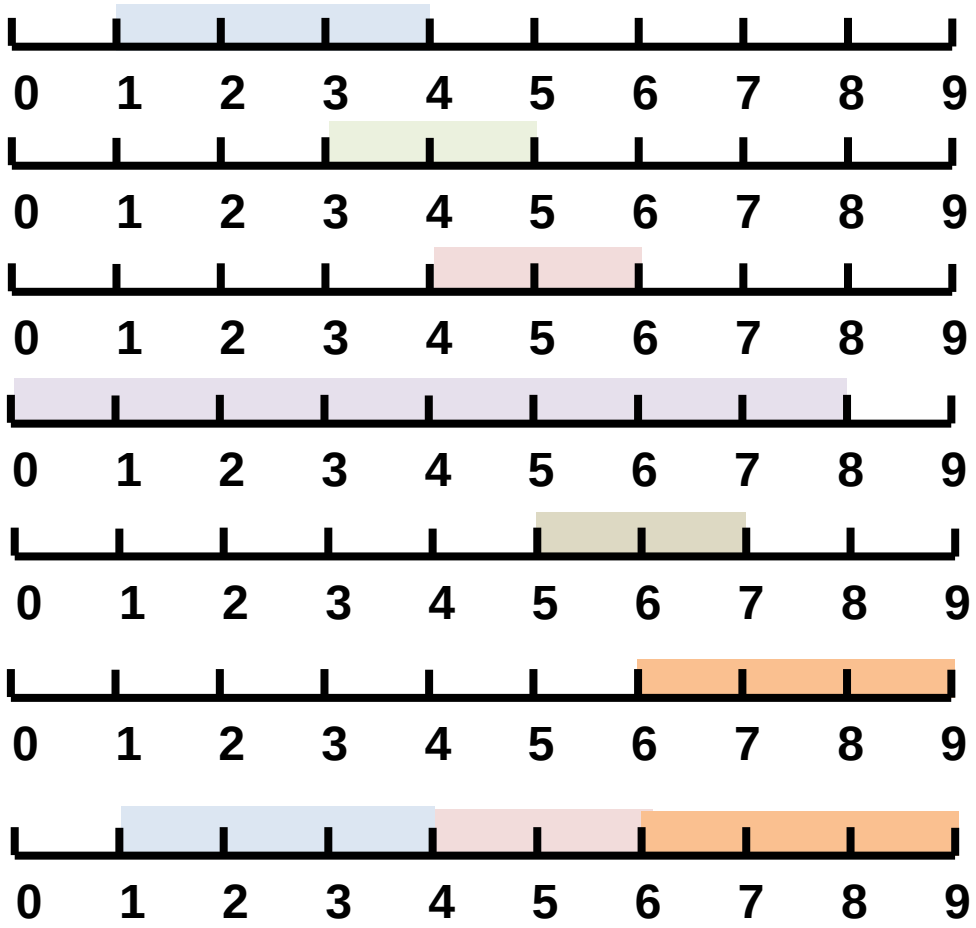
输入样例：

```
6
1 4
3 5
4 6
0 8
5 7
6 9
```

输出多少？

输出样例：

```
3
```

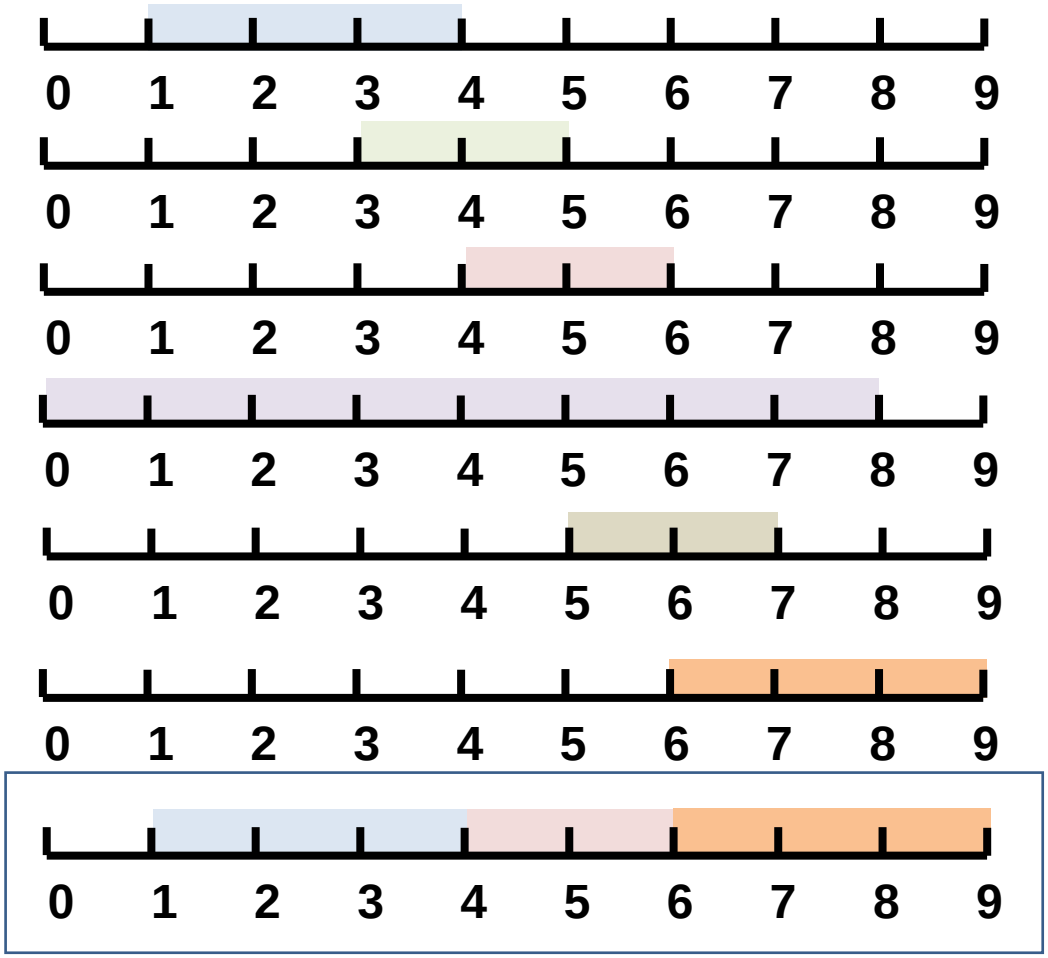


完善程序

给出n个开区间（begini, endi）
需选择**尽量多**的区间
保证两两不重叠

将活动按照endi的顺序排序，
依次考虑各活动，若未与已选
活动冲突就选，否则不选

反证贪心：假设首选最优选项
不是end1而是endi。若两者时
段不交叉，则可多选一个活动。
若交叉了则将endi换成end1，
不会使后续选择结果更差



3分钟

完善程序

```
#include<iostream>
using namespace std;
int n, begin[1009], end[1009];
void init() {
    cin >> n;
    for (int i=1;i<=n;i++) cin>>begin[i]>>end[i];
}
void qsort(int x, int y) {
    int i, j, mid, t;
    i = ____(1)____; j = y;
    mid = ____(2)____;
    while(i <= j) {
        while(end[i] < mid) i++;
        while(____(3)____)  j--;
        if(i <= j) {
            t = end[j]; end[j] = end[i]; end[i] = t;
            t = begin[j]; ____(4)____; begin[i] = t;
            i++; j--;
        }
    }
    if(x < j) qsort(x, j);
    if(i < y) qsort(i, y);
}
void solve() {
    int ans = 0;
    for(int i = 1, t = -1; i <= n; i++)
        if(begin[i]>=____(5)____) {ans++; t=end[i];}
    cout << ans << endl;
}
int main() {
    init(); qsort(1, n); solve();
    return 0;
}
```

1	识别变量
常见变量名 翻译循环变量 根据变量名的英文推断	

2	找出关键语句
控制结构(for, if) 常见算法的基本操作 函数参数、返回值	

3	理解代码段作用
翻译解释代码段	

完善程序

解释变量的作用

```
#include<iostream>
using namespace std;
int n, begin[1009], end[1009];
void init() {
    cin >> n;
    for (int i=1;i<=n;i++) cin>>begin[i]>>end[i];
}
void qsort(int x, int y) {
    int i, j, mid, t;
    i = ____(1)____; j = y;
    mid = ____(2)____;
    while(i <= j) {
        while(end[i] < mid) i++;
        while(____(3)____)  j--;
        if(i <= j) {
            t = end[j]; end[j] = end[i]; end[i] = t;
            t = begin[j]; ____(4)____; begin[i] = t;
            i++; j--;
        }
    }
    if(x < j) qsort(x, j);
    if(i < y) qsort(i, y);
}
void solve() {
    int ans = 0;
    for(int i = 1, t = -1; i <= n; i++)
        if(begin[i]>=____(5)____) {ans++; t=end[i];}
    cout << ans << endl;
}
int main() {
    init(); qsort(1, n); solve();
    return 0;
}
```

n	活动总数输入
begin[i]	第i项活动起始时间
end[i]	第i项活动的结束时间
x, y	本层快排的左右边界编号
i, j	快速排序的双游标
mid	本层快排的基准值pivot
t	上一个被选活动结束时间
ans	可以安排的最多活动数

完善程序

代码段作用

```
#include<iostream>
using namespace std;
int n, begin[1009], end[1009];
void init() {
    cin >> n;
    for (int i=1;i<=n;i++) cin>>begin[i]>>end[i];
}
void qsort(int x, int y) {
    int i, j, mid, t;
    i = ____(1)____; j = y;
    mid = ____(2)____;
    while(i <= j) {
        while(end[i] < mid) i++;
        while(____(3)____)  j--;
        if(i <= j) {
            t = end[j]; end[j] = end[i]; end[i] = t;
            t = begin[j]; ____(4)____; begin[i] = t;
            i++; j--;
        }
    }
    if(x < j) qsort(x, j);
    if(i < y) qsort(i, y);
}
void solve() {
    int ans = 0;
    for(int i = 1, t = -1; i <= n; i++)
        if(begin[i]>=____(5)____) {ans++; t=end[i];}
    cout << ans << endl;
}
int main() {
    init(); qsort(1, n); solve();
    return 0;
}
```

1：输入各活动起始结束时间

2：按endi顺序对活动快速排序

3：排好序后贪心地安排活动

1.初始化 2.排序 3.贪心安排

完善程序

```
#include<iostream>
using namespace std;
int n, begin[1009], end[1009];
void init() {
    cin >> n;
    for (int i=1;i<=n;i++) cin>>begin[i]>>end[i];
}
void qsort(int x, int y) {
    int i, j, mid, t;
    i = ____(1)____; j = y;
    mid = ____(2)____;
    while(i <= j) {
        while(end[i] < mid) i++;
        while(____(3)____)  j--;
        if(i <= j) {
            t = end[j]; end[j] = end[i]; end[i] = t;
            t = begin[j]; ____(4)____; begin[i] = t;
            i++; j--;
        }
    }
    if(x < j) qsort(x, j);
    if(i < y) qsort(i, y);
}
void solve() {
    int ans = 0;
    for(int i = 1, t = -1; i <= n; i++)
        if(begin[i]>=____(5)____) {ans++; t=end[i];}
    cout << ans << endl;
}
int main() {
    init(); qsort(1, n); solve();
    return 0;
}
```

关键语句

1：输入各活动起始结束时间

2：按endi顺序对活动快速排序

左右游标初始化为本层边界

选取一个endi设为本层基准

游标定位到首对左右互换位

左右坑位互换值并移动游标

按游标位置分别对左侧小区和右侧大区递归进行**分区**

3：排好序后贪心地安排活动

排序后顺序遍历每个活动i，只要起始时间>=上轮结束时间则安排它，并更新最新结束时间

1.初始化 2.排序 3.贪心安排

完善程序

```
#include<iostream>
using namespace std;
int n, begin[1009], end[1009];
void init() {
    cin >> n;
    for (int i=1;i<=n;i++) cin>>begin[i]>>end[i];
}
void qsort(int x, int y) {
    int i, j, mid, t;
    i = ____(1)____; j = y;
    mid = ____(2)____;
    while(i <= j) {
        while(end[i] < mid) i++;
        while(____(3)____)  j--;
        if(i <= j) {
            t = end[j]; end[j] = end[i]; end[i] = t;
            t = begin[j]; ____(4)____; begin[i] = t;
            i++; j--;
        }
    }
    if(x < j) qsort(x, j);
    if(i < y) qsort(i, y);
}
void solve() {
    int ans = 0;
    for(int i = 1, t = -1; i <= n; i++)
        if(begin[i]>=____(5)____) {ans++; t=end[i];}
    cout << ans << endl;
}
int main() {
    init(); qsort(1, n); solve();
    return 0;
}
```

(1) x

(2) end[(x+y)/2]

(3) end[i]>mid

(4) begin[i]=begin[j]

(5) t

完善程序

完善程序

（细胞问题）一个矩形阵列，由数字0到9组成。数字1到9是代表细胞。细胞的定义是，沿细胞数字上下左右四个方向，只要还是非零的细胞数字，则被视为同一个细胞，求给出的矩形阵列的**细胞总个数**。
程序在第一行输入两个数字，分别表示矩阵的行数m和列数n，接下来m行，每行n个个位数字，表示整个m*n矩阵的情况。

手算样例

输入样例：

```
4  10
0234500067
1034560500
2045600671
0000000089
```

输出多少？

输出样例：

```
4
```

```
4  int direction_x[4] = _____(1)_____;
5  int direction_y[4] = {0, 1, 0, -1};
6  int flag[100][100], m, n, num_of_cells = 0;
7  void find(int start_x, int start_y) {
8      int t=0, w=1, queue[1000][2], flag[start_x][start_y]=0;
9      num_of_cells++; t = 0; w = 1;
10     queue[1][1]=start_x;  queue[1][2]=start_y;
11     do {
12         t++;
13         for (int i = 0; i <= 3; i++) {
14             int x=queue[t][1]+direction_x[i];
15             int y=____(2)____;
16             if((x>=0)&&(x<m)&&(y>=0)&&(y<=m)&&(flag[x][y])) {
17                 w++; queue[w][1]=x; queue[w][2]=y; flag[x][y]=0;
18             }
19         }
20     }while(____(3)____);
21 }
22 int main(){
23     int i, j;  string s;  cin >> m >> n;
24     for(i = 0; i <= m - 1; i++)
25         for(j = 0; j <= n - 1; j++) flag[i][j] = 1;
26     for(i = 0; i <= m - 1; i++) {
27         cin >> s;
28         for(j=0;j<=n-1;j++) if(s[j]=='0') ____(4)____;
29     }
30     for(i = 0; i <= m - 1; i++)
31         for(j=0;j<=n-1;j++) if(flag[i][j]) ____(5)____;
32     cout<<"NUMBER of cells is "<< num_of_cells << endl;
33     return 0;
```

3分钟

1	识别变量
常见变量名 翻译循环变量 根据变量名的英文推断	

2	找出关键语句
控制结构(for, if) 常见算法的基本操作 函数参数、返回值	

3	理解代码段作用
翻译解释代码段	

解释变量作用

```
4   int direction_x[4] = _____(1)_____;
5   int direction_y[4] = {0, 1, 0, -1};
6   int flag[100][100], m, n, num_of_cells = 0;
7   void find(int start_x, int start_y) {
8       int t=0, w=1, queue[1000][2], flag[start_x][start_y]=0;
9       num_of_cells++; t = 0; w = 1;
10      queue[1][1]=start_x;  queue[1][2]=start_y;
11      do {
12          t++;
13          for (int i = 0; i <= 3; i++) {
14              int x=queue[t][1]+direction_x[i];
15              int y=____(2)____;
16              if((x>=0)&&(x<m)&&(y>=0)&&(y<=m)&&(flag[x][y])) {
17                  w++; queue[w][1]=x; queue[w][2]=y; flag[x][y]=0;
18              }
19          }
20      }while(____(3)____);
21  }
22  int main(){
23      int i, j;  string s;  cin >> m >> n;
24      for(i = 0; i <= m - 1; i++)
25          for(j = 0; j <= n - 1; j++) flag[i][j] = 1;
26      for(i = 0; i <= m - 1; i++) {
27          cin >> s;
28          for(j=0;j<=n-1;j++) if(s[j]=='0') ____(4)____;
29      }
30      for(i = 0; i <= m - 1; i++)
31          for(j=0;j<=n-1;j++) if(flag[i][j]) ____(5)____;
32      cout<<"NUMBER of cells is "<< num_of_cells << endl;
33      return 0;
```

m	行数：第0行到第m-1行
n	列数：第0列到第n-1列
flag[i][j]	标记(i,j)位是否是细胞
游标t	标识队列中当前访问位
游标w	队尾，新点入队则加1
queue[][1]	存放队列中各点行号
queue[][2]	存放队列中各点列号
num_of_cells	连通块数量

```
4   int direction_x[4] = ____(1)____;
5   int direction_y[4] = {0, 1, 0, -1};
6   int flag[100][100], m, n, num_of_cells = 0;
7   void find(int start_x, int start_y) {
8       int t=0, w=1, queue[1000][2], flag[start_x][start_y]=0;
9       num_of_cells++; t = 0; w = 1;
10      queue[1][1]=start_x;  queue[1][2]=start_y;
11      do {
12          t++;
13          for (int i = 0; i <= 3; i++) {
14              int x=queue[t][1]+direction_x[i];
15              int y=____(2)____;
16              if((x>=0)&&(x<m)&&(y>=0)&&(y<=m)&&(flag[x][y])) {
17                  w++; queue[w][1]=x; queue[w][2]=y; flag[x][y]=0;
18              }
19          }
20      }while(____(3)____);
21  }
22  int main(){
23      int i, j;  string s;  cin >> m >> n;
24      for(i = 0; i <= m - 1; i++)
25          for(j = 0; j <= n - 1; j++) flag[i][j] = 1;
26      for(i = 0; i <= m - 1; i++) {
27          cin >> s;
28          for(j=0;j<=n-1;j++) if(s[j]=='0') ____(4)____;
29      }
30      for(i = 0; i <= m - 1; i++)
31          for(j=0;j<=n-1;j++) if(flag[i][j]) ____(5)____;
32      cout<<"NUMBER of cells is "<< num_of_cells << endl;
33      return 0;
```

设置连通四方向数组d_x和d_y

从(start_x, start_y)开始
通过一个二维数组模拟的队列
用BFS发洪水算法搜索并标记

输入矩阵并标记flag数组

统计矩阵中四方向连通块数量

关键语句

```
4   int direction_x[4] = _____(1)_____;
5   int direction_y[4] = {0, 1, 0, -1};
6   int flag[100][100], m, n, num_of_cells = 0;
7   void find(int start_x, int start_y) {
8       int t=0, w=1, queue[1000][2], flag[start_x][start_y]=0;
9       num_of_cells++; t = 0; w = 1;
10      queue[1][1]=start_x;  queue[1][2]=start_y;
11      do {
12          t++;
13          for (int i = 0; i <= 3; i++) {
14              int x=queue[t][1]+direction_x[i];
15              int y=____(2)____;
16              if((x>=0)&&(x<m)&&(y>=0)&&(y<=m)&&(flag[x][y])) {
17                  w++; queue[w][1]=x; queue[w][2]=y; flag[x][y]=0;
18              }
19          }
20      }while(____(3)____);
21  }
22  int main(){
23      int i, j;  string s;  cin >> m >> n;
24      for(i = 0; i <= m - 1; i++)
25          for(j = 0; j <= n - 1; j++) flag[i][j] =
26      for(i = 0; i <= m - 1; i++) {
27          cin >> s;
28          for(j=0;j<=n-1;j++) if(s[j]=='0') ____(4)____;
29      }
30      for(i = 0; i <= m - 1; i++)
31          for(j=0;j<=n-1;j++) if(flag[i][j]) ____(5)____;
32      cout<<"NUMBER of cells is "<< num_of_cells << endl;
33      return 0;
```

从(start_x, start_y)开始搜索
通过一个二维数组模拟的队列
用BFS发洪水算法访问并标记

1:定义二维数组queue模拟队列
2:初始化当前访问位游标t和队尾w
3:将起始点flag标记为0并加入队列
4：连通块统计量num_of_cells加1

游标t++移动到应访问的队首点
利用d_x和d_y访问目前点**四方向点**

如果该邻居点还在矩阵范围内且是
未访问过的细胞点(x,y) 则加入**队列**

只要队列中还有元素（t<w）
整个BFS搜索通过while持续进行
遍历完整个连通块并取消它们标记

统计矩阵中四方向连通块数量

关键语句

```
4   int direction_x[4] = ____(1)____;
5   int direction_y[4] = {0, 1, 0, -1};
6   int flag[100][100], m, n, num_of_cells = 0;
7   void find(int start_x, int start_y) {
8       int t=0, w=1, queue[1000][2], flag[start_x][start_y]=0;
9       num_of_cells++; t = 0; w = 1;
10      queue[1][1]=start_x;  queue[1][2]=start_y;
11      do {
12          t++;
13          for (int i = 0; i <= 3; i++) {
14              int x=queue[t][1]+direction_x[i];
15              int y=____(2)____;
16              if((x>=0)&&(x<m)&&(y>=0)&&(y<=m)&&(flag[x][y])) {
17                  w++; queue[w][1]=x; queue[w][2]=y; flag[x][y]=0;
18              }
19          }
20      }while(____(3)____);
21  }
22  int main(){
23      int i, j;  string s;  cin >> m >> n;
24      for(i = 0; i <= m - 1; i++)
25          for(j = 0; j <= n - 1; j++) flag[i][j] = 1;
26      for(i = 0; i <= m - 1; i++) {
27          cin >> s;
28          for(j=0;j<=n-1;j++) if(s[j]=='0') ____(4)____;
29      }
30      for(i = 0; i <= m - 1; i++)
31          for(j=0;j<=n-1;j++) if(flag[i][j]) ____(5)____
32      cout<<"NUMBER of cells is "<< num_of_cells << endl;
33      return 0;
```

{-1, 0, 1, 0}

queue[t][2]+direction_y[i]

t<w

0

find(i, j)