

## 第六章 提高级 CSP-S 新题型初赛模拟试题

### 提高级 CSP-S 第 1 套初赛模拟试题

一、单项选择题(共 15 题,每题 2 分,共计 30 分;每题有且仅有一个正确选项)

1. 大多数计算机病毒主要造成计算机( )的损坏。  
A. 软件和数据  
B. 硬件和数据  
C. 硬件、软件和数据  
D. 硬件和软件
2. 假设今年中秋有 253 个月饼,把它们装到 15 个盒子里面,那么数量最多的一盒至少装几个月饼( )。  
A. 16  
B. 23  
C. 15  
D. 17
3. ASCII 编码是由美国国家标准委员会指定的一种包括数字、字母、通用字符和控制符号在内的字符编码集,它是一种( )位二进制编码。  
A. 8  
B. 7  
C. 4  
D. 32
4. 计算机的硬件主要包括控制器、( )、存储器、输入设备、输出设备。  
A. 运算器  
B. 操作系统  
C. 计算机语言  
D. 磁盘
5. 字符“a”的 ASCII 码是 97,写出下面程序的输出结果:char c='a'+4;cout<<c<<"", "<<(int) c+3<<endl; ( )。  
A. e, h  
B. 101, 104  
C. e, 104  
D. 101, h
6. 操作系统是对( )进行管理的软件。  
A. 计算机资源  
B. 软件  
C. 硬件  
D. 应用程序
7. 以下选项中( )不是一个操作系统环境。  
A. Linux  
B. Windows CE  
C. Solaris  
D. Celeron;
8. 以下关于 C++语言注释的说法正确的是( )。  
A. 写 C++语言程序时必须书写注释,否则会对程序的功能造成影响  
B. C++语言的注释将参与编译器编译,并形成指令  
C. 可以采用/\*……\*/的形式书写多行注释,其中的注释内容可以是任何字符  
D. //注释表示从//开始直到本行末尾的所有字符均是注释内容
9. 要使用 putchar 函数实现向显示器输出字符“A”,则可以使用( )。  
A. putchar(65)  
B. putchar(A)  
C. putchar('\65')  
D. putchar("A")
10. 两个指针( )。  
A. 可在一定条件下相加  
B. 如果同时指向一个变量,则此后就不能再指向其他变量了  
C. 任何时候都不能相减  
D. 可在一定条件下进行相等或不等的比较运算
11. 下列属于 B 类 IP 地址的是( )。  
A. 27.33.119.2  
B. 134.300.12.4  
C. 133.201.189.32  
D. 192.97.32.121
12. 现有变量 a, b, c, d, 取值范围均为[0, 15], 假设每个值出现的概率相同, 则表达式  $a \oplus b \oplus c \oplus d$  的值能被 3 整除的概率( )。(⊕为计算机中的异或运算符,结果用分数形式表达)

- A.  $\frac{3}{8}$       B.  $\frac{1}{2}$       C.  $\frac{1}{4}$       D.  $\frac{1}{8}$

13. 假设以 S 和 X 分别表示进栈和出栈操作, 则对输入序列 a, b, c, d, e 进行一系列栈操作 SSXSXSXXX 之后, 得到的输出序列为( )。

- A. baced      B. bceda      C. cbaed      D. edcba

14. 某递归算法的执行时间的递推关系如下:

当  $n=1$  时,  $T(n)=1$ ;

当  $n>1$  时,  $T(n)=2 \times T(n/2)+1$ 。

则该算法的时间复杂度为( )。

- A.  $O(1)$       B.  $O(\log_2 n)$       C.  $O(n)$       D.  $O(n * \log_2 n)$

15. 一棵完全二叉树中有 501 个叶子节点, 则至少有( )个节点。

- A. 501      B. 502      C. 1001      D. 1002

二、阅读程序(程序输入不超过数组或字符串定义的范围, 判断题正确的打√, 错误的打×; 除特殊说明外, 判断题每题 1.5 分, 选择题每题 4 分, 共计 40 分)

1.

```
01 #include<iostream>
02 using namespace std;
03
04 const int maxn=100001;
05
06 int N,M,K;
07 int x[maxn],y[maxn],d[maxn];
08 int c[maxn];
09 int*a[maxn];
10
11 int main() {
12     cin>>N>>M>>K;
13     for (int i=0;i<K;++i) {
14         cin>>x[i]>>y[i]>>d[i]; //表示第 x[i] 行第 y[i] 列的值为 d[i]
15         c[y[i]]++;
16     }
17     for (int i=1;i<=M;++i)
18         a[i]=new int[c[i]];
19     for (int i=0;i<K;++i) {
20         *a[y[i]]=d[i];
21         a[y[i]]++;
22     }
23
24     for (int i=1;i<=M;++i) {
25         a[i]=a[i]-c[i];
26         for (int j=0;j<c[i];++j,++a[i])
27             cout<<*a[i]<<' ';
28     }
29     return 0;
30 }
```



## ●判断题

- (1) 程序第 09 行定义了一个指针数组,  $a[i]$  表示第  $i$  列的指针。( )  
 (2) 第 20 行代码改成  $a[y[i]][0]=d[i]$  不影响运算结果。( )  
 (3) 第 15 行中, 数组  $c$  用来统计每行中的数据个数。( )  
 (4) 在本程序中, 采用动态数组以优化空间的利用, 每一列数组长度可能不同。( )

## ●选择题

- (5) 该程序的时间复杂度为( )。  
 A.  $O(M * N * K)$  B.  $O(M+K)$   
 C.  $O(M+N)$  D.  $O(K)$   
 (6) 该程序的空间复杂度为( )。  
 A.  $O(M+K)$  B.  $O(N * K)$  C.  $O(M+N)$  D.  $O(M * N)$

## 2.

```

01 #include<iostream>
02 #include<iomanip>
03 using namespace std;
04
05 int m[101][101];
06
07 int main() {
08     int a;
09     cin>>a;
10
11     int c=a*a,i=1,k=(a+1)/2;
12     for (int j=1;j<=c;j++) {
13         m[i][k]=j;
14         if (j%a==0) {
15             if (i==a)
16                 i=1;
17             else
18                 i++;
19         } else {
20             if (i==1)
21                 i=a;
22             else
23                 i--;
24
25             if (k==a)
26                 k=1;
27             else
28                 k++;
29         }
30     }
31     for (int i=1;i<=a;i++) {
32         for (int j=1;j<=a;j++)
33             cout<<setw(5)<<m[i][j];
34         cout<<endl;

```

```

35     }
36     return 0;
37 }

```

●判断题

(1)从程序可以看出,  $i$  为被填数,  $j$  和  $k$  为填数位置。( )

(2)填数结束后, 数组  $m$  中的元素互不相同。( )

●选择题

(3)当  $j\%a==0$  且  $i!=a$  时, 下一步填入的是( )。

A.  $m[1][k]$       B.  $m[i+1][k]$       C.  $m[k+1][i]$       D.  $m[k+1][i+1]$

(4)当  $j\%a!=0, i!=1$  且  $k==a$  时, 下一步填入的是( )。

A.  $m[a][1]$       B.  $m[i-1][1]$       C.  $m[a][k+1]$       D.  $m[i-1][k+1]$

(5)填数后, 每行每列及对角线的和均为( )。

A.  $\frac{(a^2+1) \times a}{2}$       B.  $\frac{(a^2+1)}{2}$       C.  $(a^2+1) \times a$       D.  $a^2+1$

3.

```

01 #include<iostream>
02 using namespace std;
03
04 int a[101], d[101];
05
06 int main() {
07     int n=5;
08     a[1]=d[1]=1;
09     for (int i=1; i<=n; ++i) {
10         int s=i+1, x=0;
11         for (int j=1; j<=n+1-i; ++j) {
12             int k=s+x;
13             x++;
14
15             a[j+1]=a[j]+k;
16             cout<<a[j]<<' ';
17         }
18         cout<<"..."<<endl;
19         a[1]=d[i+1]=d[i]+i;
20     }
21     return 0;
22 }

```

●判断题

(1)该题有两重循环构成, 外循环  $i$  控制列的变化, 内循环  $j$  是控制行的变化。( )

```

1 3 6 10 15
2 5 9 14
4 8 13
7 12
11

```

(2)这段代码的运行结果是 11。( )

●选择题

(3)本题在输出时, 每行为( )个  $a[j]$  数组的值。

A.  $n+1-i$       B.  $n+1$       C.  $n+1+i$       D.  $n$



(4) 本题代码的运算结果是输出( )行。

A. 4

B. 5

C. 6

D. 7

### 三、完善程序(单选题, 每小题 3 分, 共计 30 分)

1. 形如  $2^p-1$  的素数称为麦森数, 这时  $P$  一定也是个素数。但反过来不一定, 即如果  $P$  是个素数,  $2^p-1$  不一定也是素数。到 1998 年底, 人们已找到了 37 个麦森数。最大的一个是  $P=3021377$ , 它有 909526 位。麦森数有许多重要应用, 它与完全数密切相关。

你的任务: 输入  $P$  ( $1000 < P < 3100000$ ), 计算  $2^p-1$  的位数和最后 500 位数字(用十进制高精度数表示)。

输入数据:

只包含一个整数  $P$  ( $1000 < P < 3100000$ )。

输出要求:

第 1 行: 十进制高精度数  $2^p-1$  的位数。第 2~11 行: 十进制高精度数  $2^p-1$  的最后 500 位数字。(每行输出 50 位, 共输出 10 行, 不足 500 位时高位补 0)

```
01 #include<stdio>
02 #include<memory>
03 #include<cmath>
04 #define LEN 125
05
06 void Multiply(int*a,int*b) {
07     int i,j;
08     int nCarry;
09     int nTmp;
10     int c[LEN];
11
12     memset(c,0,sizeof(int)*LEN);
13     for (i=0;i<LEN;i++) {
14         nCarry=0;
15         for (j=0; ① ;j++) {
16             nTmp=c[i+j]+a[j]*b[i]+nCarry;
17             c[i+j]=nTmp%10000;
18             nCarry=nTmp/10000;
19         }
20     }
21     memcpy(a,c,LEN*sizeof(int));
22 }
23
24 int main() {
25     int i;
26     int p;
27     int anPow[LEN];
28     int aResult[LEN];
29
30     scanf("%d",&p);
31     printf("%d\n", (int) (p*log10(2))+1);
32     anPow[0]=2;
```

```

33     aResult[0]=1;
34     for (i=1;i<LEN;i++) {
35         anPow[i]=0;
36         aResult[i]=0;
37     }
38     while ( ② ) {
39         if ( ③ )
40             Multiply(aResult,anPow);
41         p>>=1;
42         Multiply(anPow,anPow);
43     }
44     aResult[0]--;
45     for (i=LEN-1;i>=0;i--) {
46         if ( ④ )
47             printf("%02d\n%02d",aResult[i]/100,aResult[i]%100);
48         else {
49             printf("%04d",aResult[i]);
50             if (i%25==0)
51                 printf("\n");
52         }
53     }
54     return 0;
55 }

```

- (1) ①处应该填( )。
- A.  $j < LEN$       B.  $j < LEN - i - 1$       C.  $j < LEN - i$       D.  $j < 1$
- (2) ②处应该填( )。
- A.  $p > 0$       B.  $p = 0$       C.  $p < 0$       D.  $p \geq 0$
- (3) ③处应该填( )。
- A.  $p \& 1$       B.  $p$       C.  $p \parallel 1$       D.  $p = 0$
- (4) ④处应该填( )。
- A.  $i \neq 0$       B.  $i > 0$       C.  $i \% 10 == 0$       D.  $i \% 25 == 12$

## 2. 问题描述:

在遥远的国家佛罗布尼亚,嫌犯是否有罪须由陪审团决定。陪审团是由法官从公众中挑选的。先随机挑选  $n$  个人作为陪审团的候选人,然后再从这  $n$  个人中选  $m$  人组成陪审团。选  $m$  人的办法:

控方和辩方会根据对候选人的喜欢程度,给所有候选人打分,分值从 0 到 20。为了公平起见,法官选出陪审团的原则是选出的  $m$  个人,必须满足辩方总分和控方总分的差的绝对值最小。如果有多种选择方案的辩方总分和控方总分之差的绝对值相同,那么选辩控双方总分之和最大的方案即可。最终选出的方案称为陪审团方案。

输入数据:

输入包含多组数据。每组数据的第一行是两个整数  $n$  和  $m$ ,  $n$  是候选人数目,  $m$  是陪审团人数。注意,  $1 \leq n \leq 200$ ,  $1 \leq m \leq 20$ , 而且  $m \leq n$ 。接下来的  $n$  行, 每行表示一个候选人的信息, 它包含 2 个整数, 先后是控方和辩方对该候选人的打分。候选人按出现的先后从 1 开始编号。两组有效数据之间以空行分隔。最后一组数据  $n = m = 0$ 。

输出要求:

对每组数据, 先输出一行, 表示答案所属的组号, 如 "Jury #1", "Jury #2", 等。接下



来的一行要象例子那样输出陪审团的控方总分和辩方总分。再下来一行要以升序输出陪审团里每个成员的编号,两个成员编号之间用空格分隔。每组输出数据须以一个空行结束。

```

01 #include<stdio>
02 #include<stdlib>
03 #include<memory>
04 #include<algorithm>
05
06 int f[30][1000];
07 int Path[30][1000];
08 int P[300];
09 int D[300];
10 int Answer[30];
11
12 int main() {
13     int i,j,k;
14     int t1,t2;
15     int n,m;
16     int nMinP_D;
17     int nCaseNo;
18     nCaseNo=0;
19
20     scanf("%d%d",&n,&m);
21     while (n+m) {
22         nCaseNo++;
23         for (i=1;i<=n;i++)
24             scanf("%d%d",&P[i],&D[i]);
25         memset(f,-1,sizeof(f));
26         memset(Path,0,sizeof(Path));
27         nMinP_D= ① ;
28         ② ;
29         for (j=0;j<m;j++) {
30             for (k=0; ③ ;k++)
31                 if ( ④ ) {
32                     for (i=1;i<=n;i++)
33                         if ( ⑤ ) {
34                             t1=j;
35                             t2=k;
36                             while (t1>0 && Path[t1][t2]!=i) {
37                                 t2--P[Path[t1][t2]]-D[Path[t1][t2]];
38                                 t1--;
39                             }
40                             if (t1==0) {
41                                 f[j+1][k+P[i]-D[i]]=f[j][k]+P[i]+D[i];
42                                 Path[j+1][k+P[i]-D[i]]=i;
43                             }

```

```

44     }
45 }
46 }
47 i=nMinP_D;
48 j=0;
49 while (f[m][i+j]<0 && f[m][i-j]<0) j++;
50 if (f[m][i+j]>f[m][i-j])
51     k=i+j;
52 else
53     k=i-j;
54 printf("Jury #%d\n",nCaseNo);
55 printf("Best jury has value%d for prosecution and value%d for
defence:\n", (k-nMinP_D+f[m][k])/2, (f[m][k]-k+nMinP_D)/2);
56 for (i=1;i<=m;i++) {
57     ⑥;
58     k-=P[Answer[i]]-D[Answer[i]];
59 }
60 std::sort(Answer+1,Answer+m+1);
61 for (i=1;i<=m;i++) printf("%d",Answer[i]);
62 printf("\n");
63 printf("\n");
64 scanf("%d%d",&n,&m);
65 }
66 return 0;
67 }

```

(1)①处应填( )。

- A.  $nMinP\_D = m * 20$   
C.  $nMinP\_D = m * 200$

- B.  $nMinP\_D = m$   
D.  $nMinP\_D = m * n$

(2)②处应填( )。

- A.  $f[0][nMinP\_D] = 1$   
C.  $f[0][nMinP\_D] > 0$

- B.  $f[0][nMinP\_D] = 0$   
D.  $f[0][nMinP\_D] > 1$

(3)③处应填( )。

- A.  $k < nMinP\_D * 2$   
C.  $k \leq nMinP\_D * 2$

- B.  $k < nMinP\_D$   
D.  $k \leq nMinP\_D$

(4)④处应填( )。

- A.  $f[j][k] > 1$       B.  $f[j][k] \geq 1$       C.  $f[j][k] \geq 0$       D.  $f[j][k] > 0$

(5)⑤处应填( )。

- A.  $f[j][k] + P[i] + D[i] > f[j+1][k+P[i]] - D[i]$   
B.  $f[j][k] + P[i] + D[i] > f[j+1][k+P[i]]$   
C.  $f[j][k] + P[i] > f[j+1][k+P[i]]$   
D.  $f[j][k] + P[i] + D[i] > f[j][k+P[i]] - D[i]$

(6)⑥处应填( )。

- A.  $Answer[i] = Path[m-i][k]$   
B.  $Answer[i] = Path[m-i][k+1]$   
C.  $Answer[i] = Path[m-i+1][k+1]$   
D.  $Answer[i] = Path[m-i+1][k]$