CSP-S 模拟题-4

（120 分钟）

1. 单项选择题（每题 2 分，15 题，共 30 分，仅有一个正确选项）

1. （ ）是一种先进先出的线性表。

A. 栈　　B. 队列　　C. 哈希表（散列表）　　D. 二叉树

2. 十六进制数 9A 在（ ）进制下是 232。

A. 四　　B. 八　　C. 十　　D. 十二

3. 如果一棵二叉树的中序遍历是 BAC，那么它的先序遍历不可能是（ ）。

A. ABC　　B. CBA

C. ACB　　D. BAC

4. 使用冒泡排序对序列进行升序排序，每执行一次交换操作将会减少 1 个逆序对，因此序列 5, 4, 3, 2, 1 需要执行（ ）次交换操作，才能完成冒泡排序。

A. 0　　B. 5　　C. 10　　D. 15

5. 如果一个栈初始时为空，且当前栈中的元素从栈底到栈顶依次为 a, b, c（如右图所示），另有元素 d 已经出栈，则可能的入栈顺序是（ ）。

A. a, d, c, b　　B. b, a, c, d　　C. a, c, b, d　　D. d, a, b, c

6. 地址总线的位数决定了 CPU 可直接寻址的内存空间大小，例如地址总线为 16 位，其最大的可寻址空间为 64KB。如果地址总线是 32 位，则理论上最大可寻址的内存空间为（ ）。

A. 128KB　B. 1MB　C. 1GB　D. 4GB

7. 蓝牙和 Wi-Fi 都是（ ）设备。

A. 无线广域网　　B. 无线城域网　　C. 无线局域网　　D. 无线路由器

8. 原字符串中任意一段连续的字符组成的新字符串称为子串。则字符串“AAABBBCCC”共有（ ）个不同的非空子串。

A. 3　　B. 12　　C. 36　　D. 45

9. 每份考卷都有一个 8 位二进制序列号。当且仅当一个序列号含有偶数个 1 时，它才是有效的。例如，00000000、01010011 都是有效的序列号，而 11111110 不是。那么，有效的序列号共有________个

A. 256　　B. 128　　C. 64　　D. 32

10. 定义字符串的基本操作为：删除一个字符、插入一个字符和将一个字符修改成另一个字符这三种操作。将字符串 A 变成字符串 B 的最少操作步数，称为字符串 A 到字符串 B 的编辑距离。字符串"ABCDEFG"到字符串"BADECG"的编辑距离为________。

A. 3　　B. 4　　C. 5　　D. 6

11. 把 M 个同样的球放到 N 个同样的袋子里，允许有的袋子空着不放，问共有多少种不同的放置方法？（用 K 表示）。例如：M=7，N=3 时，K=8；在这里认为（5,1,1）和（1,5,1）是同一种放置方法。问：M=8，N=5 时，K=________。

A. 18　　B. 36　　C. 12　　D. 24

12. 【NOIP2000 普及组】在待排序的数据表已经为有序时，下列排序算法中花费时间反而多的是( )

A. 堆排序　　B. 希尔排序

C. 冒泡排序　　D. 快速排序

13. 【NOIP2006】某个车站呈狭长形，宽度只能容下一台车，并且只有一个出入口。已知某时刻该车站状态为空，从这一时刻开始的出入记录为：“进，出，进，进，进，出，出，进，进，进，出，出”。假设车辆入站的顺序为 1，2，3，…，则车辆出站的顺序为( )。

A. 1，2，3，4，5　　B. 1，2，4，5，7

C. 1，4，3，7，6　　D. 1，4，3，7，2

14. 【NOIP2005】二叉树 T 的宽度优先遍历序列为 A B C D E F G H I，已知 A 是 C 的父结点：D 是 G 的父节点，F 是 I 的父节点，树中所有结点的最大深度为 3（根节点深度设为 0）可知 E 的父结点是( ).

A. B　　B. C

C. D　　D. E

15. 【NOIP2004 普及组】某大学计算机专业的必修课及其先修课程如下表所示：

课程代号	C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7
课程名称	高等数学	程序设计语言	离散数学	数据结构	编译技术	操作系统	普通物理	计算机原理
先修课程			C_0,C_1	C_1,C_2	C_3	C_3,C_7	C_0	C_6

请你判断下列课程安排方案哪个是不合理的( )。

A. $C_0,C_6,C_7,C_1,C_2,C_3,C_4,C_5$　　B. $C_0,C_1,C_2,C_3,C_4,C_6,C_7,C_5$

C. $C_0,C_1,C_6,C_7,C_2,C_3,C_4,C_5$　　D. $C_0,C_1,C_6,C_7,C_5,C_2,C_3,C_4$　　E. $C_0,C_1,C_2,C_3,C_6,C_7,C_5,C_4$

2. 阅读程序题（程序输入不超过数组或字符串定义的范围；判断题正确填√，错误填×；判断题 1.5 分，选择题 3 分，共计 40 分）

1.

```
#include<iostream>
using namespace std;

const int maxn=100001;

int N,M,K;
int x[maxn],y[maxn],d[maxn];
int c[maxn];
int*a[maxn];

int main(){
 cin>>N>>M>>K;
 for (int i=0;i<K;++i) {
     cin>>x[i]>>y[i]>>d[i];//表示第 x[i]行第 y[i]列的值为 d[i]
     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。

对　　　　　　　　错

2. 第 20 行代码改成 a[y[i]][0]=d[i]不影响运算结果。

对　　　　　　　　错

3. 第 15 行中, 数组用来统计每行中的数据个数。

对　　　　　　　　错

4. 在本程序中, 采用动态数组以优化空间的利用, 每一列数组长可能不同。

对　　　　　　　　错

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、

```
#include<iostream>
#include<iomanip>
using namespace std;

int m[101][101];

int main() {
    int a;

    cin>>a;

    int c=a*a,i=1,k=(a+1)/2;
    for (int j=1;j<=c;j++) {
        m[i][k]=j;
    if (j%a==0) {
        if (i==a)
            i=1;
        else
            i++;
    }
    else {
        if (i==1)
```

```
            i=a;
        else
            i--;

        if (k==a)
            k=1;
        else
            k++;
        }
    }
    for (int i=1;i<=a;i++) {
        for(int j=1;j<=a;j++)
            cout<<setw(5)<<m[i][j];
        cout<<endl;
    }
    return 0;
}
```

判断题：

1. 从程序可以看出，i 为被填数，j 和 k 为填数位置。

对　　　错

2. 填数结束后，数组 m 中的元素互不相同。

对　　　错

选择题：

3. 当 j%a==0 且 i!=0 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!=0 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.

```
#include <iostream>
using namespace std;

int a[101],d[101];

int main()
{
    int n=5;
    a[1]=d[1]=1;
    for (int i=1;i<=n;++i) {
        int s=i+1,x=0;
        for (int j=1;j<=n+1-i;++j){
            int k=s+x;
            x++;

            a[j+1]=a[j]+k;
            cout<<a[j]<<' ';
        }
        cout<<"..."<<endl;
        a[1]=d[i+1]=d[i]+i;
```

```
    }
    return 0;
}
```

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。

对　　　　　　错

2. 该段代码的运行结果是

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

。

对　　　　　　错

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

3. 完善程序题（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: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 500 位时高位补 0)。

```
#include<cstdio>
#include<memory>
#include<cmath>
#define LEN 125

void Multiply(int*a,int*b) {
    int i,j;
    int nCarry;
    int nTmp;
    int c[LEN];

    memset(c,0,sizeof(int)*LEN);
    for (i=0;i<LEN;i++){
        nCarrv=0;
        for (j=0;_____①_____;j++) {
            nTmp=c[i+j]+a[j]*b[i]+nCarry;
            c[itj]=nTmp%10000;
            nCarry=nTmp/10000;
        }
```

```
    }
    memcpy(a,c,LEN*sizeof(int));
}
int main() {
    int i;
    int p;
    int anPow[LEN];
    int aResult[LEN];

    scanf("n%d",&p);
    printf("%d\n",(int)(p*log10(2))+1);
    anPow[0]=2;
    aResult[0]=1;
    for (i=1;i<LEN;i++) {
        anPow[i]=0;
        aResult[i]=0;
    }
    while (_____②_____) {
        if (_____③_____)
            Multiply(aResult,anPow);
        p>>=1;
        Multiply(anPow,anPow);
    }
    aResult[0]--;
    for (i=LEN-1;i>=0;i--) {
        if (_____④_____)
            printf("%02d\n%02d",aResult[i]/100,aResult[i]%100;
        else {
            printf("%04d",aResult[i]);
            if (i%25==0)
                printf("\n");
        }
    }
    return 0;
}
```

1. ①处应填( )

A. j<LEN　　B. j<LEN-i-1　　C. j<LEN-i　　D. j<1

2. ②处应填()

A. p>0　　B. p==0　　C. p<0　　D. p>=0

3. ③处应填( )

A. p&1　　B. p　　C. p||1　　D. p=0

4. ④处应填( )

A. i!=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≤n≤200，1≤m<20，而且 m<=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于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。

```
#include<cstdio>
#include<memory>
#include<cstdlib>
#include<algorithm>

int f[30][1000];
int Path[30][1000];
int P[300];
int D[300];
int Answer[30];

int main() {
    int i,j,k;
    int t1,t2;
    int n,m;
    int nMinP_D;
    int nCaseNo;
    nCaseNo=0;

    scanf("%d%d",&n,&m);
    while (n+m) {
        nCaseNo++;
        for (i=1;i<=n;i++)
            scanf("%d%d",&P[i],&D[i]);
        memset(f;-1,sizeof(f));
        memset(Path,0,sizeof(Path));
        nMinP_D=_____①_____;
        _____②_____;
        for (j=0;j<m;j++) {
            for (k=0;_____③_____;k++)
                if(_____④_____)
                    for (i=1;i<=n;i++)
                        if(_____⑤_____) {
                            t1=j;
                            t2=k;
                            while (t1>0 && Path[t1][t2]!=i) {
                                t2-=P[Path[t1][t2]]-D[Path[t1][t2]];
                                t1--;
```

```
                        }
                        if (t1==0) {
                            f[j+1][k+P[i]-D[i]]=f[j][k]+P[i]+D[i];
                            Path[j+1][k+P[i]-D[i]]=i;
                        }
                    }
        }
        i=nMinP_D;
        j=0;
        while (f[m][i+j]<0 && f[m][i-j]<0) j++;
        if (f[m][i+j]>f[m][i-j])
            k=i+j;
        else
            k=i-j;
        printf("Jury #%d\n",nCaseNo);
    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);
        for (i=1;i<=m;i++) {
            ______⑥______;
            k-=P[Answer[i]]-D[Answer[i]];
        }
        std::sort(Answer+1,Answer+m+1);
        for (i=1;i<=m;i++) printf("%d",Answer[i]);
        printf("\n");
        printf("\n");
        scanf("%d%d",&n,&m);
    }
    return 0;
}
```

1. ①处应填( )

A. nMinP_D=m*20　　B. nMinP_D=m　　C. nMinP_D=m*200　　D. nMinP_D=m*n

2. ②处应填()

A. f[0][nMinP_D]=1　　B. f[0][nMinP_D=0　　C. f[0][nMinP_D]>0　　D. f[0][nMinP_D]>1

3. ③处应填( )

A. k<nMinP_D*2　　B. k<nMinP_D　　C. k<=nMinP_D*2　　D. k<=nMinP_D

4. ④处应填( )

A. f[j][k]>1　　B. f[j][k]>=1　　C. f[j][k]>=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]