提高组模拟题第一套试题及答案

1．大多数计算机病毒主要造成计算机（ A ）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

解析：

病毒破坏的主要是软件和数据。

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（ D ）

A.16　　B.23　　C.15　　D.17

解析：鸽巢原理也叫抽屉原理，m个物体放到n个抽屉里至少有一个抽屉有 $\left\lceil \frac{m}{n} \right\rceil$ 个物体。$\left\lceil \frac{253}{15} \right\rceil$=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（ B ）位二进制编码。

A.8　　B.7　　C.4　　D.32

解析：ASCII 是一种 7 位的二进制编码，能表示 2^7=128 种国际通用的西文字符，是目前计算机中，尤其是微型计算机中使用最普遍的字符编码集。

4．计算机的硬件主要包括控制器、（ A ）、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

解析：运算器是计算机主要的硬件之一，主要进行各种算术运算和逻辑运算。

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;（ C ）。

A.e,h　　B.101,104　　C.e,104　　D.101,h

解析：

char c='a'+4，c 里面保存的是 e。int(c)+3=101+3=104

6．操作系统是对（ D ）进行管理的软件。

A.计算机资源　　B.软件　　C.硬件　　D.应用程序

解析：

操作系统主要是对应用程序进行管理的软件。

7．以下选项中（ D ）不是一个操作系统环境。

A.Linux　　B. Windows CE　　C.Solaris　　D.Celeron;

解析：

Celeron 是指处理器而非操作系统环境。

8．以下关于C＋＋语言注释的说法正确的是（　D　）。

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解析：

Celeron是指处理器而非操作系统环境。

【解析】写C＋＋语言程序时可写注释，也可不写，注释只起注解说明的作用，不会参与编译、运行过程。“/*·······*/”用于注释代码块，已经注释的内容不能再次被注释，即不允许嵌套。所以C其中的注释内容可以是任何字符这句话是错误的的。不能嵌套写/*/**/*/

D“//”适用于短小精简的注释，其只能注释一行，如果要注释多行就要写多次。

9．要使用putchar函数实现向显示器输出字符“A”，则可以使用（　A　）。

A.putchar(65)　　B.putchar(A)

C.putchar('\65')　D.putchar("A")

解析：

putchar函数的作用是向显示终端输出一个字符，一般形式为putchar（c），其中参数c为整型，输出的字符是c值对应的ASCII码。

10．两个指针（　D　）。

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

解析：

如果两个指针都指向同一个数组中的元素，则可以相减，其值为两个指针之间的元素个数，但它们不能相加。某个时刻两个指针同时指向了一个变量，不影响之后改变其值而指向其他变量。

11．下列属于B类IP地址的是（　C　）。

A.27.33.119.2　　　B.134.300.12.4

C.133.201.189.32　　D.192.97.32.121

解析：

B类IP地址的范围是128.0.0.0到191.255.255.254。B选项中IP地址值超过255，故错误。

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＋b＋c+d的值能被3整除的概率（　A　）。(+为计算机中的异或运算符，结果用分数形式表达)

A.3/8　　B.1/2　　C.1/4　　D.1/8

解析：

可以试着分别计算结果为0,3,6,9,12,15的概率，注意到异或值取到每个值的概率是一样的（因为

C_4^0+c_4^2 +C_4^4 =C_4^1 +C_4^3)，因此概率是 6/16=3/8。

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 SSXSX SSXXX 之后，得到的输出序列为（　B　）。

A.baced　　B.bceda　　C.cbaed　　D.edcba

解析：

a 入栈 b 入栈 b 出栈 c 入栈 c 出栈 d 入栈 e 入栈 e 出栈 d 出栈 a 出栈

14．某递归算法的执行时间的递推关系如下：

当 n=1f 时，T(n)=1;

当 n>1 时，T(n)=2xT(n/2)+1。

则该算法的时间复杂度为（　C　）。

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

解析：

设 $n=2^k$, $k=log_2 n$

$T(n)=2^1\times T(\frac{n}{2^1})+1=2^2\times T(\frac{n}{2^2})+1+2^1=\ldots=2^k\times T(\frac{n}{2^k})+1+2^1+\ldots+2^{k-1}=2^k\times T(1)$ $+\ 2^k-1=2n-1=O(n)$。

15．一棵完全二叉树中有 501 个叶子节点，则至少有（　C　）个节点。

A.501　　B.502　　C.1001　　D.1002

解析：

n0=n2+1 这里 n0=501，所以 n2=500,n=n0+n1+n2=1001+n1,n1 为 0 或者 1，所以有 n≥1001

阅读程序

```
#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]]++;
    }
    for(int i=1;i<=M;++i)
        a[i]=new int[c[i]];
    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    }
```

（1） 程序第 9 行定义了一个指针数组，a[i]表示第 i 列的指针。()

答案 √

（2）第 20 行代码改成 a[y[i]][0]=d[i] 不影响运算结果。()

答案 √

解析：允许将指针当作数组名使用。

（3）第 15 行中，数组 c 用来统计每行中的数据个数。()

答案 ×

解析：数组 c 用来统计每列的数据个数。

（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。()

答案 √

解析：由于在本题中，N×M 可能会很大，直接用二维数组可能会很大，因而采用了动态数组和指针，根据每一列的实际数据个数来申请该列的空间，使每列的“数组”长度不同。

（5）该程序的时间复杂度为（ B ）。

A.O(M*N*K)　　B.O(M+K)

C.O(M+N)　　D.O(K)

解析：

程序的时间复杂度为 O（M＋K)。

（6）该程序的空间复杂度为（ A ）。

A.O(M+K)　　B.O(N*K)　　C.O(M+N)　　D.O(M*N)

解析：

该程序的空间复杂度为 O（M＋K)。

(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;
}
```

（1） 从程序可以看出，i 为被填数，j 和 k 为填数位置。()

答案 ×

解析： j 为被填数，i 和 k 为填数位置，即将 j 填入数组 m[i][k]中。

（2） 填数结束后，数组 m 中的元素互不相同。()

答案 √

解析：

```
for(int j=1;j<=c;j++){
        m[i][k]=j;
```

循环到 a*a,所以每次填数都不一样。

（3）当 j%a==0 且 i!=a 时，下一步填入的是（ B ）。

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

解析：符合这两个条件，执行 i++;所以是给 m[i+1][k]填数。

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

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

解析：执行了 i--; k=1; 所以填入的是 m[i-1][1]

（5）填数后，每行每列及对角线的和均为（ A ）。

A. $((a^2+1)\times a)/2$　B. $((a^2+1))/2$　C. $(a^2+1)\times a$　D. a^2+1

解析：每行每列及对角线的数的和为 $((a^2+1)\times a)/2$，即为 a 阶幻方。

(3)

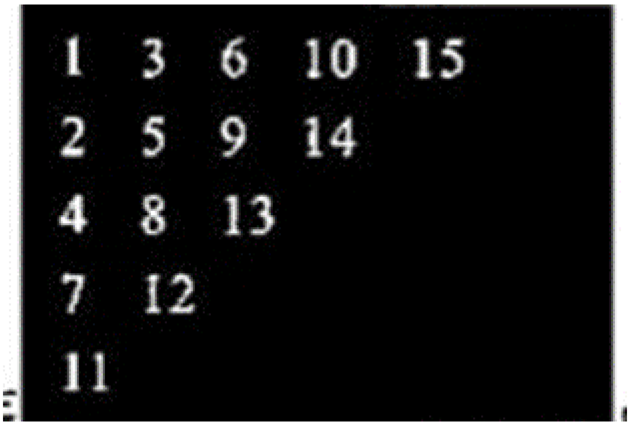
```
#include<iostream>
using namespace std;

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;
}
```

（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。（ ）

答案 ×

解析：外循环 i 控制行的变化，内循环 j 是控制列的变化。

（2）这段代码的运行结果是 。（ ）



答案 ×

解析：每行都要输出 cout<<"..."<<endl;

（3）本题在输出时，每行为（ A ）个 a[j]数组的值。

A.n+1-i　　B.n+1　　C.n+1+i　　D.n

解析：

每行有 n+1-i 个 a[j]数组的值，即第一行　(i=1)有 5+1-1 个数，第二行有 5+1-2 个数，·······，第五行有 5＋1-5 个数。

（4）本题代码的运算结果是输出（　　B　　）行。

A.4　　B.5　　C.6　　D.7

解析：

这段代码的运行结果是输出 5 行。

完善程序

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=302137777，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）

```
#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++){
             nCarry=0;
             for(j=0;___ ①____;j++){
                nTmp=c[i+j]+a[j]*b[i]+nCarry;
                c[i+j]=nTmp% 10000;
                nCarry=nTmp/10000;
             }
        }
        memcpy(a,c,LEN * sizeof(int));
   }

```

```
int main(){
    int i;
    int p;
    int anPow[LEN];
    int aResult[LEN];

    scanf("%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处应该填(　　C　　)。

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

解析：

j 只要算到 LEN-i-1，是因为 b[j]×a[j]的结果总是加到 c[i+j]上，i+j 大于等于 LEN 时，c[i+j]是不需要的，也不能要，否则 c 数组就越界了。

(2)②处应该填(　　A　　)。

A.p>0　　B.p==0　　C. p<0　　D.p>=0

解析：

p=0 则说明 p 中的有效位都用过了，不需再算下去。

(3)③处应该填(　　A　　)。

A.p&1　　B.p　　C.p111　　D.p=0

解析：

判断此时 p 中最低位是否为 1。

(4)④处应该填(　D　)。

A.i!=0　　B.i>0　　C.i%10==0　　Di%25==12

解析：

输出从万进制数的第124位开始，万进制数的每一位输出为十进制数的4位，每行只能输出50个十进制位，所以发现当i％25＝12时，第i个万进制位会被折行输出，其对应的后两个十进制位会跑到下一行。

(2)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:

输人包含多组数据。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。

```
#include <cstdio>
#include<cstdlib>
# include<memory>
#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));
    nMinPD=______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　 ）。

A. nMinP_D=m*20　　　B. nMinP_D=m

C. nMinP_D=m*200　　D. nMinP_D=m*n

解析：

辩控差为 m*20

（2）②处应填（　B　）。

A. f[0][nMinP_D]=1　　B. f[0][nMinP_D]=0

C. f[0][nMinP_D]>0　　D. f[0][nMinP_D]>1

解析：

选 0 个人使得辩控差为 nMinP_D 的方案，其辩控和就是 0。

（3）③处应填（　C　）。

A. k<nMinP_D*2　　B. k<nMinP_D

C. k<=nMinP_D*2　　D. k<=nMinP_D

解析：

可能的辩控差的范围是[0,nMinP_D*2]，故选 C。

（4）④处应填（　C　）。

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

（5）⑤处应填（　A　）。

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]]

解析：

在方案 f（j，k）的基础上，通过加进第 i 个人所形成的方案 f(j+1,k+P[i]-D[i]))，是到目前为止，选 j＋1 个人，使得辩控差为 k＋P[i]-D[i]的所有方案中最优的。

（6）⑥处应填（　D　）。

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]

解析：

最终方案 f(m，k)的最后一个人选记录在 Path[m][k]中，从 Path[m][k] 出发，找出方案 f（m，k）的所有人选，放入 Answer 数组。