普及组模拟题第十套试题及答案

1．在C＋＋中使用cin和cout应该调用（　A　）库。

A.iostream　　B.cstdio　　C.cmath　　D.stack

解析：常识题

2．n是一个三位数，那n的十位数为（　D　）。

A.(n%10)/10　　B.(n/100)%10　　C.(n/100)%100　　D.(n%100)/10

解析:

n%100 拿到n的后两位数，再/10去掉最后一位数。即得到n的十位数。

例如，364%100=64，64/10=6。注意n为整型。

3．已知大写字母A的ASCII编码为65（十进制），则大写字母J的十进制ASCII编码为（　D　）。

A.71　　B.72　　C.73　　D.74

解析：

A~J共10个字母，64+10=74。

4．计算机用户可以根据需要安装软件，那么计算机的软件系统一般分为（　A　）。

A.系统软件和应用软件　　B.管理软件和控制软件

C.军用软件和民用软件　　D.高级软件和一般软件

解析：计算机软件系统分为系统软件和应用软件。

5．一片容量为8GB的SD卡能存储大约（　C　）张大小为2MB的数码照片。

A.1600　　B.2000　　C.4000　　D.16000

解析：

8*1024/2=4096 大约存储4000张

6．一个字节(byte)由（　A　）个二进制位组成。

A.8　　B.16　　C.32　　D.以上皆有可能

解析：

一个字节=8bit组成

7．前缀表达式“＋ 3 × 2 ＋ 5 12”的值是（　C　）。

A.23　　B.25　　C.37　　D.65

解析：

从右向左运算，遇见运算符，就按照这个运算符把后面两个数运算，+ 5 12 结果是17，运算式变为 + 3 × 2 17，再计算2×17=34，运算式变为+ 3 34 最后结果是37。

8．一个字长为8位的整数的补码是1111 1001，则它的原码是（　D　）。

A.0000 0111　　B.0111 1001　　C.1111 1001　　D.1000 0111

解析：

源码到补码计算，因为第一位是 1，所以是负数。原码-》反码-》补码。

负数的补码就是除符号位意外，0 变 1，1 变 0.负数的补码就是+1。反过来计算，即可。

补码-》反码-》源码，1111 1001-》1111 1000-》1000 0111

9．基于比较的排序时间复杂度的下限（最优）是（ B ），其中 n 表示待排序的元素个数。

A.O(n) B.O(nlogn) C.O(logn) D.O(n2)

【解析】

快速排序和归并排序，时间复杂度 O(nlogn)

10．一棵二叉树的前序遍历序列是 ABCDEFG，后序遍历序列是 CBFEGDA。则根结点的左子树的结点个数可能是（ A ）。

A.2 B.3 C.4 D.5

解析：

首先前序遍历顺序是：根节点→左子树→右子树，而后序遍历顺序是：左子树→右子树→根节点，首先知 A 是根节点，又由后序遍历知 D 必然是右子树的根节点，D 前面的 ABC 中 A 是根节点，剩下的 B、C 两个节点必然是左子树的，答案是 2 个。

11．十进制小数 13.375 对应的二进制数是（ A ）。

A.1101.011 B.1011.011 C.1101.101 D.1010.01

解析：

13=2*6 余 1 6=2*3 余 0 3=2*1 余 1 1=2*0 余 1

整数部分 1101

0.375*2=0.75 0.75*2=1.5 0.5*2=1.0

小数部分 0.011

二进制数：1101.011

12．根据域名代码规定，表示政府部门网站的域名代码是（ B ）。

A..net B..gov C..com D..org

解析：

常识题，单看英文简写也知道。.gov 是政府部门网站。.org 是非营利性组织。

13．计算机中的数值信息分为整数和实数（浮点数）。实数之所以能够表示很大或者很小的数，是由于使用了（ A ）。

A.阶码 B.补码 C.反码 D.较长的尾数

解析：因为使用了阶码，在机器中表示一个浮点数时需要给出指数，这个指数用正数形式表示，这个正数叫做阶码。阶码指明了小数点在数据中的位置。

14．计划展出 10 幅不同的画，其中 1 幅水彩画、4 幅油画、5 幅国画，排成一行陈列，要求同一品种的画必须连在一起，并且水彩画不放在两端，那么不同的陈列方式有（ D ）种。

A.2880　　B.17280　　C.8640　　D.5760

解析：4幅油画有A_4^4种不同的排法；5幅国画有A_5^5种不同的排法；水彩画放在油画和国画之间，则有$24\times120\times2=5760$种不同的陈列方法。

15．定义一种字符串操作，一次可以将其中一个元素移到任意位置。举例说明，对于字符串“BCA”可以将“A”移到“B”之前，变字符串“ABC”。如果要将字符串“DACHEBCIF”变成“ABCDEFGHI”最少需要（　A　）次操作。

A.4　　B.5　　C.6　　D.7

解析：

ACDHEBGIF→ABCDHEGIF→ABCDHEFGI→ABCDEFGHI，4 次就可以完成。

阅读程序

```
1 #include<bits/stdc++.h>
2 using namespace std;
3 const int maxn=50;
4 void getnext(char str[]){
5     int l=strlen(str),i,j,k,temp;
6     k=l-2;
7     while(k-1>=0&&str[k]>str[k+1])k--;
8     i=k+1;
9     while(i<l&& str[i]>str[k])i++;
10         temp=str[k];
11         str[k]=str[i-1];
12         str[i-1]=temp;
13         for(i=l-1;i>k;i--)
14         for(j=k+1;j<i;j++)
15            if(str[j]>str[j+1]){
16               temp=str[j];
17               str[j]=str[j+1];
18               str[j+1]=temp;
19          }
20    return;
21  }
22   int main(){
23  char a[maxn];
24  int n;
25  cin>>a>>n;
26  while(n>0){
27      getnext(a);
28      n--;
29   }
30   cout<<a<<endl;
31   return 0;
32   }
```

（1）若输入的字符串 a 是升序的，那么无论 n 为多少，第 13 行的循环都不会执行（执行的条件不

满足)。(　　)

答案 ×

解析：若输人的字符串 a 是升序的，那么无论 n 为多少，第 13 行的循环都会执行。

（2）若输入的 n 等于 2，对于第 27 行，第一次执行完和第二次执行完的字符串最多只有 2 个字符位置不同。(　　)

答案 ×

解析：比如 12345 执行完第一次是 12354，第二次执行完是 12435，改变的字符多于 2 个。

（3）程序执行完第 7 行时，第 k＋1 个字符到第 l-1 个字符的值是不严格递减的。()

答案 √

解析：由“while(k-1>=0&&str[k]>str[k+1]) k--;”知，第 k＋1 个字符到第 l-1 个字符的值是不严格递减的。

（4）程序执行完第 12 行时，第 k＋1 个字符到第 l-1 个字符的值是不严格递减的。()

答案 √

解析： str_k 和 str_(i-1)交换后，也不会改变第 k+1 个字符到第 l-1 个字符的值是不严格递减的。

（5）若输入的字符串有 x 个字符并且是严格降序的，输入的 n 等于 1，则第 16~18 行会执行（　A　）次。

A.(x-1)(x-2)/2　　B.x(x-1)/2　　C.x(x+1)/2　　D.0

解析：

x-2+x-1+・・・+1=(x-1)(x-2)/2。

（6）若输入的字符串有 x 个字符并且都相同，输入的 n 等于 1，则第 16~18 行会执行（　D　）次。

A.(x-1)(x-2)/2　　B.x(x-1)/2　　C.x(x+1)/2　　D.0

解析：

x 个字符都相同，对于任意 j,str_j >str_(j+1)不成立，则不会运行第 16～18 行。

(2)

```
1 #include<bits/stdc++.h>
2 using namespace std;
3 int w[35000],d[35000],dp[35000];
4 int main()
5{
6    int n,m;
7    scanf("%d%d",&n,&m);
8    for(int i=1;i<=n;++i)
9         scanf("%d%d",&w[i],&d[i]);
10   for(int i=1;i<=n;++i)
11   {
12        for(int j=m;j>=w[i];--j)
```

```
13            dp[j]=max(dp[j],dp[j-w[i]]+d[i]);
14    }
15    printf("%d\n",dp[m]);
16    return 0;
17 }
```

（1）上述代码中，双重循环里循环变量 j 的枚举顺序改为从 W[i] 到 m，输出结果一定不变。

(　　)

答案 ×

解析： j 的枚举顺序改为从 W[i] 到 m，就变成了完全背包。因为 01 背包再更新的时候数组的值会被提前更新，所以才从后往前枚举。

（2）上述代码中，双重循环中变量 i 的枚举顺序改为从 n 到 1，输出结果一定不变。（　　）

答案 √

解析：枚举背包物品，顺序改变不会改变。

（3）若输入数据中，1<=n<=30000,1<=m<=30000,1<=w[i]<=30000,1<= d[i]<=30000，则所求答案一定没有溢出。（）

答案 √

解析：答案不会有溢出。

（4）若输入数据中，1<=n<=30000,1<=m<=30000,1<=w[i]<=30000,1<= d[i]<=10^9，则所求答案一定没有溢出。（ ）

答案 ×

解析： dp[j-w[i]]+d[i]，所以 d[i]的值到了 1 亿，有可能溢出。

（5）当输入为：

4 6

1 4

2 6

3 12

2 7

输出为（ D ）。

A.17　　B.28　　C.29　　D.23

解析：

选 1 4， 2 7， 3 12 这三件物品，最大价值 23。

		0	1	2	3	4	5	6
0		0	0	0	0	0	0	0
1	1，4	0	4	4	4	4	4	4

2	2，6	0	4	10	10	10	10	10
3	3，12	0	4	10	12	16	22	22
4	2，7	0	4	10	12	17	19	23

（6）上述代码的时间复杂度为（ C ）。

A.O(n)　　B.O(n2m)　　C.O(nm)　　D.O(n ㎡)

解析：

时间复杂度是 O(n*m),m 是背包容量。

(3)

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

（1）将第 7 行“i=1”改为“i=0”，程序不会出错。（）

答案 ×

解析：结果会出错，输入 6，递归 r(n-i)，如果 i 是 0，死循环。

（2）程序输出的结果有可能小于-1。（）

答案 √

解析：输入的数 n<=NUM ，直接 return n.所以输入一个负数，就返回一个负数，肯定比-1 小。

（3）若程序两次输入的值分别为 n1 和 n2，且有 n1-n2=1 的关系，则对于这两次运行的结果 ans1 和 ans2，有 ans1-ans2=1。（）

答案 ×

解析：结果是-6，因为输入 6，结果是-1，输入 5 结果是 5。

（4）若输入的 n 大于等于 6 时，程序一定至少执行一次第 9 行。（）

答案 √

解析： 6 的倍数都会执行一次第 9 行。

（5）（2 分）若输入 2020，输出的结果为（ B ）。

A.3 B.4 C.5 D.-1

解析：

2020%6=4，返回结果为 4。

（6）若已知 0<=n<=100，则要使输出的结果为-1，则 n 的取值有（ D ）种。

A.10 B.12 C.14 D.16

解析：

6 的倍数的个数大约 16 个。

完善程序

1.（过河问题）在一个月黑风高的夜晚，有一群人在河的右岸，想通过唯一的一根独木桥走到河的左岸。在伸手不见五指的黑夜里，过桥时必须借照灯光来照明，不幸的是，他们只有一盏灯。另外，独木桥上最多能承受两个人同时经过，否则将会坍塌。每个人单独过独木桥都需要一定的时间，不同的人用的时间可能不同。两个人一起过独木桥时，由于只有一盏灯，所以需要的时间是较慢的那个人单独过桥所花费的时间。现在输入 N（2<=N<1000）和这 N 个人单独过桥需要的时间，请计算总共最少需要多少时间，他们才能全部到达河左岸。

例如，有 3 个人甲、乙、丙，他们单独过桥的时间分别为 1、2、4，则总共最少需要的时间 为 7。具体方法是：甲、乙一起过桥到河的左岸，甲单独回到河的右岸将灯带回，然后甲、丙在一起过桥到河的左岸，总时间为 2+1+4=7。

```
1   #include<iostream>
2   #include<cstring>
3   using namespace std;
4   const int SIZE=100;
5   const int INFINITY=10000;
6   const bool LEFT=true;
7   const bool RIGHT=false;
8   const bool LEFT_TO_RIGHT=true;
9   const bool RIGHT_TO_LEFT=false;
10 int n,hour[SIZE];
11bool pos[SIZE];
12int max(int a,int b)
13  {
14      if(a>b)
15            return a;
16      else
17            return b;
18   }
19   int go(bool stage)
20   {
21     int i,j,num,tmp,ans;
22     if(stage==RIGHT_TO_LEFT)
23     {
```

```
    num=0;
    ans=0;
    for(i=1;i<=n;++i)
          if(pos[i]==RIGHT)
          {
            num++;
            if(hour[i]>ans)
                ans=hour[i];
          }
    if( ___①_____)
       return ans;
    ans=INFINITY;
    for(i=1;i<=n-1;++i)
          if(pos[i]==RIGHT)
             for(j=i+1;j<=n;++j)
        if(pos[j]==RIGHT)
        {
             pos[i]=LEFT;
             pos[j]=LEFT;
             tmp= max(hour[i],hour[j])+ ____②____
             if(tmp<ans)
                ans=tmp;
             pos[i]=RIGHT;
             pos[j]=RIGHT;
          }
    return ans;
  }
  if(stage ==LEFT_TO_RIGHT)
  {
    ans=INFINITY;
    for(i=1;i<=n;++i)
         if(___③___)
         {
           pos[i]=RIGHT;
           tmp= ____④____;
           if(tmp <ans)
               ans=tmp;
               _____⑤__;
         }
     return ans;
 }
return 0;
}
int main()
{
 int i;;
 cin>>n;
 for(i=1;i<=n;++i)
 {
   cin>>hour[i];
   pos[i]=RIGHT;
```

```
75  }
76  cout<<go(RIGHT_TO_LEFT)<<endl;
77  return 0;
78 }
```

（1）①处应填（ C ）。

A.num<=0　　B.num<=1　　C.num<=2　　D.num<=3

解析：

当 num<=2 时，右边人全都走到左边任务就结束了。

（2）②处应填（ C ）。

A.go(RIGHT)　　B.go(!pos[i])

C.go(LEFT_TO_RIGHT)　　D.go(RIGHT_TO_LEFT)

解析：

从右岸走过去后，一个人还必须从左岸走回来，所以调用 go(LEFT_TO_ RIGHT)。

（3）③处应填（ A ）。

A.pos[i]=LEFT　　B.pos[i]==RIGHT

C.num<=2　　D.num<=3

解析：

先判断 pos[i]==LEFT，从左岸走过来。

（4）④处应填（ B ）。

A.hour[i]+go(LEFT_TO_RIGHT)　　B.hour[i]+go(RIGHT_TO_LEFT)

C.go(LEFT_TO_RIGHT)　　D.go(RIGHT_TO_LEFT)

解析：从左岸走过去后，还必须从右岸走回来，所以调用 go(RIGHT_TO_LEFT)，则 hour[i]+go(RIGHT_TO_LEFT）就是花费的时间。

（5）⑤处应填（ D ）。

A.return ans　　B.tmp=0　　C.pos[i]=RIGHT　　D.pos[i]=LEFT

解析：

递归后，状态改回来，pos[i]=LEFT。

2.（国王放置）在 n×m 的棋盘上放置 k 个国王要求 k 个国王互相不攻击，有多少种不同的放置方法？假设国王放置在第（x,y）格，国王的攻击区域是（x-1,y-1),(x-1,y),(x-1,y+1),(x,y-1),(x,y+1),(x+1,y-1),(x+1,y)，(x+1,y+1)。读入三个数 n，m， k，输出答案。题目利用回溯法求解。棋盘行标号为 0~n-1，列标号为 0~m-1。试补全程序。

```
1 #include <cstdio>
2 #include <cstring>
3 int n,m,k,ans;
4 int hash[5];
5 void work(int x, int y, int tot){
6   int i,j;
7   if(tot==k){
```

```
8       ans++;
9       return;
10  }
11  do{
12          while(hash(hash[x][y]){
13          y++;
14          if(y==m){
15               x++;
16               y=____①___;
17          }
18          if(x==n)
19          return;
20      }
21      for(i=x-1;i<=x+1;i++)
22        if(i>=0&&i<n)
23          for(j=y-1;j<=y+1;j++)
24                          if(j>=0&&j<m)
25                          _____②__;
26              ______③___;
27              for(i=x-1;i<=x+1;i++)
28                if(i>=0&&i<n)
29                  for(j=y-1;j<=y+1;j++)
30                   if(j>=0&&j<m)
31                     hash[i][j]--;
32              ___④__ ;
33             if(y==m){
34              x++;
35              y=0;
36             }
37            if(x==n)
38            return;
39       }while(1);
40  }
41  int main(){
42  scanf("%d%d%d",&n,&m,&k);
43  ans=0;
44  memset(hash,0,sizeof(hash));
45  ____⑤____;
46  printf("%d\n",ans);
47  return 0;
48  }
```

（1）①处应填（ A ）。

A.0　　B.1　　C.x　　D.x+1

解析：

当 y=m 时，y 从 0 开始重新侦测。

（2）②处应填（ B ）。

A.hash[i][j]=1　　B.hash[i][j]++

C.hash[i][j]=0　　D.hash[i][j]—

解析：

与第 31 行对称，hash[i][j]++表示已放置过国王。

（3）③处应填（　D　）。

A.work(x,y+1,tot+1)　　B.y--

C.y++　　D.work(x,y,tot+1)

解析：

从 (x,y)再继续搜索，并且放置国王的数量已经＋1。

（4）④处应填（　D　）。

A.y=0　　B.y=1　　C.y--　　D.y++

解析：

y++，表示可以进行下一列的侦测。

（5）⑤处应填（　C　）。

A.work(n-1,m-1,0)　　B.work(n,m,0)

C.work(0,0,0)　　D.work(1,1,0)

解析：

从 0 行 0 列开始搜索，初始放置国王数量为 0。