提高组模拟题第六套试题及答案

1．下列 C++＋表达式，值最大的是（　B　）。

A.'Z'-'A'　　B.52%53>>1

C.(rand()-rand()+1)%26　　D.20+15%28/3

解析：

‘Z’-‘A’ ASCII 码相减结果是 25

52%53>>1 算数运算符运算级别高于左移右移。52>>1 结果 26

(rand()-rand()+1)%26 小于等于 25 的数

20+15%28/3　20+5 25

所以选 B，26 最大。

2．下列属于解释执行的程序设计语言是（　D　）。

A.C　　B.C++　　C. Pascal　　D. Python

解析：

解释执行是编译一行执行一行。

编译执行是一下子全部编译好后执行。

python 就是解释执行的程序设计语言。

3．对于 64KB 的存储器，其最大地址码用十六进制表示是（　B　）。

A.10000　　B.FFFF　　C. 1FFFF　　D. EFFFF

解析：

64KB=64*1024B 也就是 64*1024 个字节，64 是 2 的 6 次方，1024 是 2 的 10 次方，

64*1024=2^6*2^10=2^16 结果是 2 的 16 次方个字节，结果是 65536。

从 0 开始就是 65535 转换 16 进制 FFFF

4．为解决 Web 应用中的不兼容问题，保障信息的顺利流通（　D　），制定了一系列标准，涉及 HTML、XML、CSS 等，并建议开发者遵循。

A.微软　　B.美国计算机协会（ACM）

C.联合国教科文组织　　D. 万维网联盟（W3C）

解析：

web 标准的制定者是万维网联盟,又称 W3C 理事会。

5．若一个整数是另一个整数的平方，则称该数是“完全平方数”，如 4、9、25 等。下列表达式无法判断该整数是否为完全平方数的是（　C　）。x 是整型。

A.floor(sqrt(x)+0.5)==sqrt(x)

B.int(sqrt(x))==sqrt(x)

C.x/int(sqrt(x))==int(x/int(sqrt(x)))

D.int(sqrt(x))* int(sqrt(x))==x

解析：

这道题需要注意 sqrt 函数的返回值是 doubble 类型。

结果转换成整型，向下取整，C 选项，假设 x=5,左边 5/2=2 右边 5/2=2,所以无法判断是否是完全

平方数。
选项 A floor 向下取整再加上 0.5 就变成四舍五入取整，如果不是完全平方数，四舍五入取整后肯定不会和一个 double 的数相等。
选项 B 跟 A 差不多，开平方后向下取整，同样如果不是完全平方数，向下取整后肯定不会和一个 double 的数相等。
D 也是同样的道理。
6．将 8 个名额分给 5 个不同的班级，允许有的班级没有名额，有几种不同的分配方案

（　C　）。

A.60　　B.120　　C.495　　D.792

解析：

同素分配问题采用隔板法，标准隔板法公式C_{n-1}^{m-1}，采用隔板法的少分型，8个名额分配5个班级，有的没有名额，可以每个班级先借走一个名额，然后每个班级至少分配一个名额，即转换成标准的隔板法。5个班级，每个班级借一个名额，共计有13个名额，就变成了13个名额，带入标准公式C_{13-1}^{5-1} =C C_{12}^{4} =495种。

7．同时查找 2n 个数中的最大值和最小值，最少比较次数为（　C　）。

A.3(n-2)/2　　B.4n-2　　C.3n-2　　D.2n-2

解析：

前两个数比较一次，大的是最大值，小的是最小值。然后后面 2*（n-1）个数，每两个比较一次，需要比较 n-1 次，两个数中的最大数还要和最大值比较一次，也是 n-1 次，两个数中的最小值也和最小值比较一次，也是 n-1 次，这样就比较了 3*（n-1）次。在加上前两个数比较一次。3*(n-1)+1=3n-3+1=3n-2。选择 C。

8．具有 n 个顶点，e 条边的图采用邻接表存储结构，进行深度优先遍历和广度优先遍历运算的时间复杂度均为（ D　　）。

A.O(n2)　　B.O(e2)　　C.O(ne)　　D.O(n+e)

解析：答案 O(n+e)邻接表的 dfs 和 bfs，时间复杂度跟顶点和边的数量有关，所以选 D。

9．两根粗细相同、材料相同的蜡烛，长度比是 21：16，它们同时开始燃烧，18 分钟后，长蜡烛与短蜡烛的长度比是 15：11，则较长的那根蜡烛还能燃烧（　A　）。
A．150 分钟　　B．225 分钟　　C．128 分钟　　D．9 分钟
解析：根据比和比例应用关系进行解答即可。
解：因为是同时燃烧，两根蜡烛原来与现在的长度差是不变的

	原来	现在	原来	现在
第一根	21	15	21×4=84	15×5=75
第二根	16	11	16×4=64	11×5=55

差	5	4	20	20

根据差不变，则说明前一个比值的 5 份等于后一个比值的 4 份，所以它们分别可以写成 21:6=84:64，15:11=75:55。看上表可知，他们的差都为 20 份，也就是说 18 分钟内，长蜡烛从 84 份燃烧到 75 份，而短蜡烛从 64 份燃烧到 55 份。那么长蜡烛还能燃烧 75×(18/(84-75))=150

10．一次数学考试试题由两部分组成，结果全班有 15 人得满分，第一部分做对的有 31 人，第二部分做错的有 19 人，那么两部分都做错的有（ A ）。

A.3　　B.4　　C.6　　D.12

解析：

因为有 15 人满分，且第一部分有 31 人做对，于是第二部分就有 15 人做对.又因为第二部分有 19 人做错。所以全班共有 15+19=34 人。所以第一部分有 34-31=3 人做错。第二部分错的 19 人，所以两部分都错的有 3 人。

11．设一组初始记录关键字序列为（50,40,95,20,15,70,60,45），则以初始增量值 d=4 进行希尔排序，第二趟结束后前 4 条记录关键字为（ A ）。

A.15,20,50,40　　B.15,40,60,20　　C.15,20,40,45　　D.15,20,40,50

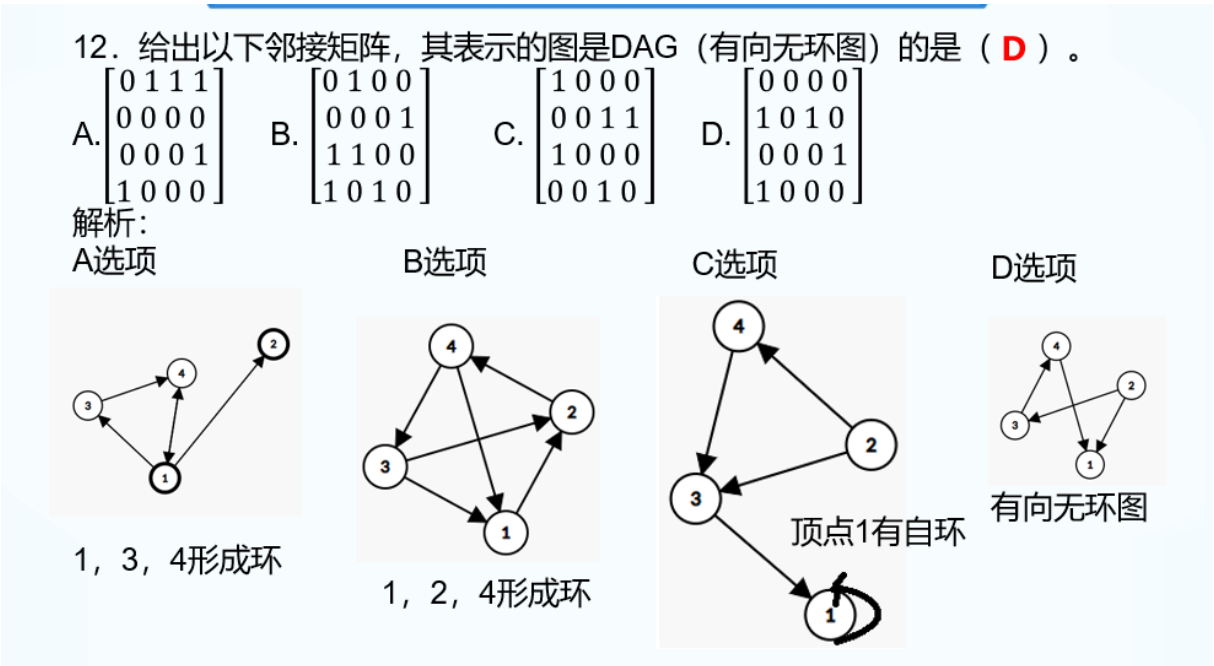
解析：

选希尔排序本质就是带增量的插入排序。本质意思为：先将整个待排元素序列由相隔的某个增量分割成若干个子序列分别进行直接插入排序，然后依次缩减增量再进行排序，当整个序列的元素基本有序时，再对全体的元素进行依次直接插入排序。故整个增量排序的变化过程为：

d=4:15,40,60,20,50,70,95,45

d=2:15,20,50,40,60,45,95,70

d=1:15,20,40,45,50,60,70,95



13．下列关于最短路算法的说法正确的是（ B ）。

A.当图中不存在负权回路但是存在负权边时，Dijkstra 算法可以求出源点到所有点的最短路

B.当图中不存在负权边时，调用多次 Dijkstra 算法能求出每对顶点间最短路径

C.当图中存在负权回路时，调用一次 Dijkstra 算法也一定能求出源点到所有点的最短路

D.当图中存在负权边时，Dijkstra+堆优化算法可以求出源点到所有点的最短路

解析：

Dijkstra 不能求负权边得最短路。但是可以调用多次求出每对顶点间的最短路径。

14．数列{a_n}是等差数列，首项 a_1>0,a_2020+a_2021>0，a_2020*a_2021<0，则使前 n 项和 s_n>0 成立的最大项数 n 是（　B　）。

A.2020　　B.4040　　C.4041　　D.4042

解析：

由等差序列性质，如果 x+y=4041，则 a_x+a_y=a_2020+a_2021>0，所以 s_4040>0 显然 a_2021<0，所以若 x+y=4042，a_x+a_y=a_2021*2<0，所以 s_4041<0。

15．给定长为 n(n≤10000）的字符串，每次可以将连续一段回文序列消去，消去后左右两边会接到一起，求最少消去几次能消完整个序列（单个字符也算回文字符串）。设 f（i，j）表示消去闭区间［i，j］内字符串所需要的最小次数，那么当 1≤i≤j≤n 时，在不考虑回文串的情况下，f(i,j）的动态规划方程中包含（　A　）。

A. $\min_{i\le k<j}\{f(i,k)+f(k+1,j)\}$　B. $\min_{i\le k<j}\{f(i,k)+f(k+1,j)\}+1$

C. $\min_{i\le k<j}\{(f(i,k)*f(k+1,j)\}$　D.f(i,k)+f(k+1,j),i≤k<j

解析：

区间 DP 题。首先第一重循环枚举区间长度，接着第二重循环枚举左端点 i，通过计算得出右端点位置 j。如果此时左端点 i 和右端点 j 处的字母相同则可以直接消掉即等于较小的区间 DP 值。注意特判相同时如果 i 与 j 相邻则值为 1，因为需消掉一次。

枚举断点 k，那么这种方案的操作数显然是 f[i][k]+f[k+1][j]，选 A。

阅读程序

```
1   #include<iostream>
2   #include<cstdio>
3   using namespace std;
4   int L,ans;
5   char a[2002][2002];
6   int cross(int x,int y){
7   int length=1;
8   if(x<=1||x>=L)return 1;
9   for(int i=1;;i++){
10      if(x-i<=0||x+i>=L+1) return length;
11      else if(a[x-i][y]!=a[x+i][y])return length;
12      else length+=2;
13      }
14  }
15  int down(int x,int y){
16  int length=1;
17  if(y<=1||y>=L)return 1;
18  for(int i=1;;i++){
```

```
19      if(y-i<=0||y+i>=L+1) return length;
20      else if(a[x][y-i]!=a[x][y+i])return length;
21      else length+=2;
22      }
23  }
24  int MAXN(int a,int b){
25      if(a>=b)return a;
26      else return b;
27}
28  int main(){
29      cin>>L;
30      for(int i=1;i<=L;i++)
31      for(int j=1;j<=L;j++)
32          cin>>a[i][j];
33      int x,y;
34      cin>>x>>y;
35      ans=MAXN(cross(x,y),down(x,y));
36      cout<<ans;
37      return 0;
38  }
```

（1）第 35 行若改成 MAXN(down(x,y),cross(x,y))，运行结果不变。（）

答案 √

解析：求最大值，在前在后无关。

（2）第 34 行输入值包含 0 时，程序可能会产生 Runtime Error。( )

答案 ×

解析： if(x<=1||x>=L)return 1;程序有特判所以不会出现 Runtime Error。

（3）程序输出的 ans 可能等于 0。( )

答案 ×

解析：子函数里面 length 初始化为 1。所以两个函数返回值的最大值不可能是 0.

（4）当第 34 行输入值 x>y 时，cross(x,y)返回值必然大于 down(x,y)返回值。( )

答案 ×

解析： cross 与 down 的返回值与输入的 a 数组的值有关，与 x，y 无必然联系，故错误。

（5）对于输入的 L＊L 的字符矩阵，ans 值最大是（ B ）。

A.L-1　　B.L　　C.2L　　D.L-2

解析：

ans 最大值为字符矩阵的长，故选 B。

（6）若输入 L=5,x=y=3,aij={{abcba},{bcdcb},{cdedc}},{bcdcb},{abcba}}，则输出是（ C ）。

A.2　　B.3　　C.5　　D.10

解析：

abcba

bcdcb

cdedc

bcdcb
abcba
对于 cross（3,3）
a[3][2]=a[3][4],a[3][1]=a[3][5],答案为 5
对于 down（3,3）
a[2][3]=a[4][3],a[1][3]=a[5][3],答案为 5
故输出 5
(2)

```
1   #include<cstdio>
2   #include <cmath>
3   const int N=1e9;
4   int isnp[50005],p[50005],pcnt;
5
6   inline void getPrime(const int n=sqrt(N)){
7       isnp[0]=isnp[1]=1;
8       for(register int i=2;i<=n;++i){
9           if(! isnp[i])p[pcnt++]=i;
10          for(register int j=0;i* p[j]<=n&&j<pcnt;++j){
11              isnp[i*p[j]]=1;
12              if(!(i%p[j]))break;
13          }
14      }
15  }
16
17  int main(){
18      getPrime();int n;
19      while(scanf("%d",& n)&&N){
20          int _sqrt=sqrt(n),ans=n;
21          for (register int j=0;p[j]<=_sqrt&&j<pcnt;++j){
22              if(n%p[j]==0){
23                  while(n % p[j]==0)n/=p[j];
24                      ans=1ll*ans*(p[j]-1)/p[j];
25              }
26          }
27          if(n!=1)ans=1ll*ans*(n-1)/n;
28          printf("%d\n",ans);
29      }
30      return 0;
31  }
```

（1）若去掉第 12 行，程序也能得到正确结果。（）

答案 √

解析：欧筛（线性筛）：第 12 行的存在保证了每个合数只会被自己最小的质因数筛到，是一个优化时间复杂度的判断语句，所以不会对程序的正确性产生影响。

（2）若去掉第 23 行，程序也能得到正确结果。（）

答案 ×

解析： if(n!=1)ans=1ll*ans*(n-1)/n;

第 23 行去掉后，会导致程序在执行 27 行时，n 的值可能不一样了，所以后面 27 行的值可能有变

化。

（3）若输入的 n≤108，则第 10 行　j<pcnt 条件可以省略。（ ）

答案 ×

解析：如果省略，那么 j 可能会访问到 p[j]=0 的位置，程序会出现除 0 错误。

（4）若输入的 $n\leq 10^8$，则第 21 行　j<pcnt 条件可以省略。（ ）

答案 √

解析：当 n≤〖10〗^8 时，_sqrt 的值一定会小于 sqrt(〖10〗^9),所以 j<pcnt 省略后，循环执行的条件 p[j]<=_sqrt 循环会结束，不影响程序正确性。

（5）当 n=504 时，输出 ans 为（ B　）。

A.288　　B.144　　C.504　　D.503

【解析】504=2^3*3^2*7，所以 ans=504*(2-1)/2*(3-1)/3*(7-1)/7=144。

（6）getPrime 函数的时间复杂度是（　A　）。

A.O(n)　　B. O(logn)　　C.O√n　　D.O(n√n)

【解析】线性筛时间复杂度去掉常数后时间复杂度 O(n)。

(3)

```
1   #include<iostream>
2   using namespace std;
3
4   const int inf=0x3f3f3f3f, N=4e5+5;
5   struct edge{
6       int to,nt;
7   }E[N<<1];
8   int cnt,n,m,rt,MX;
9   int H[N],sz[N],mxs[N];
10  void add_edge(int u,int v){
11      E[++cnt]=(edge){v,H[u]};
12      H[u]=cnt;
13  }
14  void dfs(int u, int fa){
15      sz[u]=1;
16      for(int e= H[u];e;e= E[e].nt){
17          int v=E[e].to;
18          if(v==fa)continue;
19          dfs(v,u);
20          sz[u]+=sz[v];
21          mxs[u]=max(mxs[u],sz[v]);
22      }
23      mxs[u]=max(mxs[u],n-sz[u]);
24      if(mxs[u]<mxs[rt])rt=u;
25      if(mxs[u]==mxs[rt]&&u<rt)rt=u;
26  }
27  int main(){
28      cin>>n;
29      rt=cnt=0;
```

```
30      for(int i=1;i<n;i++){
31          int u,v;
32          cin>>u>>v;
33          add_edge(u,v);
34          add_edge(v,u);
35      }
36      mxs[0]=inf;
37      dfs(1,0);
38      cout<<rt<<' '<<mxs[rt]<<'\n';
39      return 0;
40  }
```

（1）第 33、34 行只需保留任意一行也不会影响程序的正确性。（）

答案 ×

解析：无向图加边，删掉一行就变成了有向图，肯定影响程序的正确性。

（2）第 37 行函数调用 dfs(x,y)，只需保证 1≤x≤n,y≤0 即可。()

答案 √

解析： x 是 dfs 的开始顶点，从所有顶点出发都可。

（3）第 32 行输入若有重复（重边），不影响输出结果的正确性。()

答案 ×

解析：没有处理重边，影响输出结果。

（4）程序运行结束时可能存在正整数 i（i≤n)使 sz[i]等于 mxs[i]。( )

答案 √

解析： n=2 时，后访问的点即满足题意。

（5）n=6，二元组（u,v）各个值分别是 {(1,3)，(6,3)，(2,6)，(5,6)，(3,4)}，则输出是

(　B　)。

A.3 2　　B.3 3　　C.6 3　　D.6 2

解析：

画出图来模拟。

完善程序

1.（翻硬币）一摞硬币共有 m 枚，每一枚都是正面朝上。取下最上面的一枚硬币，将它翻面后放回原处。然后取下最上面的 2 枚硬币，将他们整体一起翻面后放回原处（如 101111 前 5 个翻面结果是 000101）。再取 3 枚，取 4 枚・・・・・・直至 m 枚。然后再从这摞硬币最上面的一枚开始，重复刚才的做法。这样一直做下去，直到这摞硬币中每一枚又是正面朝上为止。例如，m 为 1 时，翻两次即可；m 为 4 时，翻 11 次；m 为 9 时，翻 80 次。

【输入】

仅有的一个数字是这摞硬币的枚数 m(0<m<1000)。

【输出】

为了使这摞硬币中的每一枚都是正面朝上所必须翻的次数。

【输入样例】

30

【输出样例】

899

```
#include<iostream>
using namespace std;
int solve(int m);
int main(){
    freopen("demo.out","w",stdout);
int m;
do{
    scanf("%d",&m);
    if(m>0&&m<1000)
        printf("%d\n",___①____);
    }while(m>0&&m<1000);
    return 0;
}

int solve(int m){
    int i,t,s=-1;//翻转的次数
    int flag;
    if(m==1)
        s=____②___;
    else{
        d=2*m+1
//确定硬币是经过偶数次翻转还是奇数次翻转
        t=2;
//表示一个 COIN 必须翻转偶数次，才能从正面继续翻回到正面
        i=1;//翻转的轮数，每轮为从 1 翻转到 m
        flag=0; //退出循环标志，翻转完成标志
        do{
            if(t==1){
                s=____③__;
                flag=1;
            }else if(__④__){
                s=i*m-1;
                flag=1;
            }else{
                t=___⑤___;
            }
            i=i+1;
        }while(! flag);
    }
    return s;
}
```

（1）①处应填（ C ）。

A.solve(m+1)　　B.solve(2*m)

C.solve(m)　　D.solve(m)+1

解析：

m 枚硬币，m 传值给子函数，开始翻转。

（2）②处应填（ C ）。

A.0　　B.1　　C.2　　D.-1

解析：

如果只有一枚硬币，翻两次就能达到目标

（3）③处应填（　A　）。

A.i*m　　B.i*m+1

C.i*m-1　　D.i*(m+1)

【解析】前后文分析 m 分为了两种情况 m==1 和 m>1。m>1 的时候我们看到结果和 t 的值相关，t 的值也分为了三种情况，t==1，（第二种情况还需要填空），t==其他的值。看 while 循环的条件 t==1 的时候肯定是输出结果。i 是翻转的轮数，每轮由 1 翻转到 m，分析选项，c 肯定不对，因为 c 是 t 等于另外一种情况的时候计算的值。纸上模拟一下情况 m=1,2 次，m=2,翻转 3 次

m=3 的时候翻转 9 次，m=4 的时候 11 次。根据这个情况来模拟，m=2 ， d=5,这个时候，i 肯定是 2。很显然，执行 i*m-1 结果是 3，m=3,d=7,i 肯定是 3，纸上模拟结果是 9，（i）3*（m）3=9,所以确定肯定填写 i*m。

（4）④处应填（　C　）。

A.t==2*m-1　　B.t==2*m+1

C.t==2*m　　D.t==2*(m+1)

【解析】根据上文分析，m=2，d=5 的时候符合这个条件，t 的初始值是 2，根据下一个空，t 的值会变化，m=2 的时候，i 肯定 2，也就是翻转 2 轮，肯定是 2*2-1。所以先看下一个空。第五个空，A 是((2*2)+1)%5 更新 t=0，肯定不对，B 是(2*2)%5=4 C 4%5+1 D2%5=2，A 和 D 肯定不对。

t 的另外一个条件是==4 或者是==5，第 4 个空只有 B 或者 C 能够得到 4 或者 5，纸上模拟

m=4,d=9,i=3,因为结果是 11，所以肯定 3*4-1。模拟一下看看怎么得到 3*4-1。

进入循环 t=2,空 5 得到上面两个选项 B，C 有可能是对的，B 选项 t=(2*2)%9=4，这时候 i==2，2*4%9==8，i++后，i 的值刚好是 3。符合上面的要求，如果是 C 选项 t=(2*2)%9+1=5,i=2;

(2*5)%9==1，i=3,结果就变成 3*4=12,所以不正确，因此我们得到第 5 个空选 B，第 4 个空就选 C。

（5）⑤处应填（　B　）。

A.(t*2+1)%d　　B.(t*2)%d

C.(t*2)%d+1　　D. t%d

【解析】根据上文分析，m=2，d=5 的时候符合这个条件，t 的初始值是 2，根据下一个空，t 的值会变化，m=2 的时候，i 肯定 2，也就是翻转 2 轮，肯定是 2*2-1。所以先看下一个空。第五个空，A 是((2*2)+1)%5 更新 t=0，肯定不对，B 是(2*2)%5=4 C 4%5+1 D2%5=2，A 和 D 肯定不对。

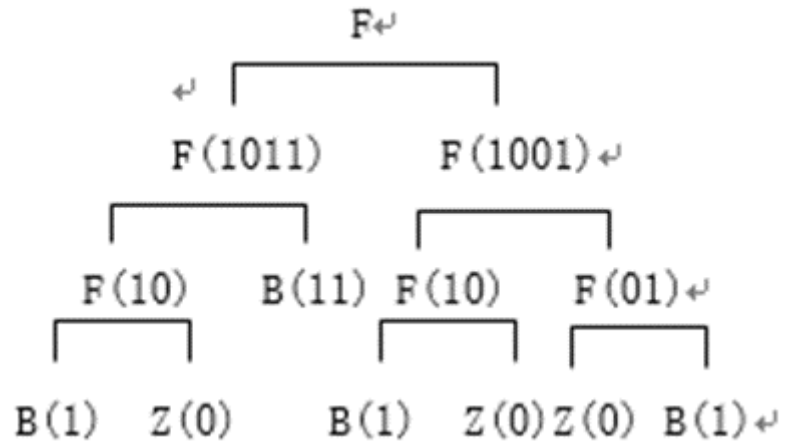
t 的另外一个条件是==4 或者是==5，第 4 个空只有 B 或者 C 能够得到 4 或者 5，纸上模拟

m=4,d=9,i=3,因为结果是 11，所以肯定 3*4-1。模拟一下看看怎么得到 3*4-1。

进入循环 t=2,空 5 得到上面两个选项 B，C 有可能是对的，B 选项 t=(2*2)%9=4，这时候 i==2，2*4%9==8，i++后，i 的值刚好是 3。符合上面的要求，如果是 C 选项 t=(2*2)%9+1=5,i=2;

(2*5)%9==1，i=3,结果就变成 3*4=12,所以不正确，因此我们得到第 5 个空选 B 。

2.（FBZ 串问题）已知一个由 0,1 字符组成的长度为 2n 的字符串。请按以下规则分解为 FBZ 串：•若其中字符全为 1，则称其为 B 串；•若其中字符全为 0，则称其为 Z 串；•若不全为 0，也不全为 1，则称 F 串。若此串为 F 串，则应将此串分解为 2 个长为 2n-1 的子串。对分解后的子串，仍按以上规则继续分解，直到全部为 B 串或为 Z 串为止。例如 n=3 时，给出 0-1 串为“10111001”



最后输出：FFFBZBFFBZFZB。给定一个 0-1 串，分解成 FBZ 串。

```
1   # include <iostream>
2   #include <cstdio>
3   #include<cstring>
4   using namespace std;
5   const int n=8;
6   char str1[2*n][n];
7   char str2[40]=" ";
8   int main(){
9       int s1,s2,x,s,t;
10      s1=s2=x=0;
11      gets(str1[0]);
12      while(___①__){
13          s=t=0;
14          for(int i=0;i<n;i++){
15              if(str1[s2][i]=='1')
16                  s++;
17              if(str1[s2][i]=='0')
18                  t++;
19          }
20          if(__②___)
21              str2[x++]='B';
22          else if(__③_)
23              str2[x++]='Z';
24          else{
25              str2[x++]='F';
26              int j=(s+t)>>1;
27              for(int k=n*2-2;k>=__④__;k--)
28                  for(int i=0;i<n;i++)
29                      str1[k+2][i]=str1[k][i];
30                  s1+=2;
31                  for(int i=0;i<j;i++){
32                      str1[s2+1][i]=str1[s2][i];
33                      str1[s2+2][i]=___ ⑤__;
34                  }
35                  for(int i=_⑥__;___⑦__;i++){
36                      str1[s2+1][i]=' ';
37                      str1[s2+2][i]=' ';
```

```
38                  }
39              }
40              s2++;
41          }
42          puts(str2);
43          return 0;
44      }
```

（1）①处应填（ B ）。

A.s1<=s2　　B.s2<=s1　　C.x<=n　　D.s<=t

【解析】s2 指当前处理的字符串，s1 指所有出现的字符串个数。

（2）②处应填（ A ）。

A.t==0　　B.s==0　　C.t>0　　D.s>0

【解析】if(str1[s2][i]==‘0’) t++；根据这条语句 t==0 就是全为 1 的 B 串。

（3）③处应填（ B ）。

A.t==0　　B.s==0　　C.t>0　　D.s>0

【解析】反之，s==0 全为 0 就是 Z 串。

（4）④处应填（ B ）。

A.s2　　B.s2+1　　C.s1　　D.s1+1

【解析】将当前的所有字符串向后移动两位。

（5）⑤处应填（ C ）。

A.str1[s2][j]　　B.str1[s2+1][i]　　C.str1[s2][i+j]　　D.str1[s2+1][i+j]

【解析】将原串分成两半。

（6）⑥处应填（ B ）。

A.1　　B.j　　C.s1　　D.s2

（7）⑦处应填（ D ）。

A.i<j　　B.i<s1　　C.i<s2　　D.i<n

【解析】将缩短的字符串后半部分清空