

2019CSP-S组 答案

2019CCF非专业级别软件能力认证第一轮 (CSP-S) 提高级C++语言试题A卷 (B卷与A卷仅顺序不同)

认证时间：2019年10月19日 上午9:30-11:30

考生注意事项：

- 丨 试题纸共有10页，答题纸共有1页，满分100分。请在答题纸上作答，写在试题纸上的一律无效
- 丨 不得使用任何电子设备（如计算器、手机、电子词典等）或查阅任何书籍资料。

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. 若有定义：int a=7; float x=2.5, y=4.7; 则表达式 $x+a\%3*(int)(x+y)\%2$ 的值是：（ ）
A.0.000000 B.2.750000 C.2.500000 D.3.500000

答案：D

解析： $x+y$ 转整数等于7， $7\%3*7\%2=1$ ，再加 x ，答案为3.5。

2. 下列属于图像文件格式的有（ ）
A.WMV B.MPEG C.JPEG D.AVI

答案：C

解析：WMV是音频格式、MPEG、AVI是视频格式、JPEG是图像格式。

3. 二进制数11 1011 1001 0111 和 01 0110 1110 1011 进行逻辑或运算的结果是（ ）
A.11 1111 1101 B.11 1111 1111 1101 C.10 1111 1111 1111 D.11 1111 1111 1111

答案：D

解析：将两个二进制数（右）对齐，逐位做或运算，每一位如果有1则或运算结果为1，14位进行或运算计算结果均为1，选D。

4. 编译器的功能是（ ）
A. 将源程序重新组合

- B. 将一种语言（通常是高级语言）翻译成另一种语言（通常是低级语言）
C. 将低级语言翻译成高级语言
D. 将一种编程语言翻译成自然语言

答案：B

解析：编译器将高级语言（例如C++，方便人创作）翻译成低级语言（机器语言，方便机器执行）。

5. 设变量x为float型且已赋值，则以下语句中能将x中的数值保留到小数点后两位，并将第三位四舍五入的是（ ）

A. $X=(x*100+0.5)/100.0$ B. $x=(int)(x*100+0.5)/100.0;$
C. $x=(x/100+0.5)*100.0$ D. $x=x*100+0.5/100.0;$

答案：B

解析：x的类型是float, 所以 $(x*100+0.5)$ 也是float, 也就是有小数位，需要先转成int, 也就是B选项。

6. 由数字1, 1, 2, 4, 8, 8所组成的不同的4位数的个数是（ ）

A.104 B.102 C.98 D.100

答案：B

解析：穷举法。1.当取出1, 1, 2, 4时，共有 $C(2,4)*2=12$ 种；2.当取出1, 1, 2, 8，也是12种；3.当取出1, 1, 4, 8，也是12种；4.当取出1, 1, 8, 8，为 $C(2,4)$ 是6种；5.当取出为1, 2, 4, 8时候，为 $A(4,4)=20$ 种；6.当取出1, 2, 8, 8，为12种；7.当取出1, 4, 8, 8为12种；8.当取出2, 4, 8, 8为12种。一共102种情况。

7. 排序的算法很多，若按排序的稳定性和不稳定性分类，则（ ）是不稳定排序。

A.冒泡排序 B.直接插入排序 C.快速排序 D.归并排序

答案：C

解析：若经过排序，这些记录的相对次序保持不变，即在原序列中， $r[i]=r[j]$ ，且 $r[i]$ 在 $r[j]$ 之前，而在排序后的序列中， $r[i]$ 仍在 $r[j]$ 之前，则称这种排序算法是稳定的。快速排序在中枢元素和 $a[j]$ 交换的时候，很有可能把前面的元素的稳定性打乱，比如序列为 5 3 3 4 3 8 9 10 11，现在中枢元素5和3(第5个元素，下标从1开始计)交换就会把元素3的稳定性打乱，所以快速排序是一个不稳定的排序算法。

8. G是一个非连通无向图（没有重边和自环），共有28条边，则该图至少有（ ）个顶点

A.10 B.9 C.11 D.8

答案：D

解析：n个点最多 $n(n+1)/2$ 条边，要不连通，至少去掉n-1条边 $n(n+1)/2-(n-1) \geq 28$ ，n最小为8。

9. 一些数字可以颠倒过来看，例如0、1、8颠倒过来看还是本身，6颠倒过来是9，9颠倒过来看还是6，其他数字颠倒过来都不构成数字。类似的，一些多位数也可以颠倒过来看，比如106颠倒过来是901。假设某个城市的车牌只有5位数字，每一位都可以取0到9。请问这个城市有多少个车牌倒过来恰好还是原来的车牌，并且车牌上的5位数能被3整除？（ ）

A.40 B.25 C.30 D.20

答案：B

解析：前2位有0,1,8,6,9，5种选择，第3位只能放0,1,8，后2位由前2位决定。而0,1,8模3正好余0,1,2，所以给定其他4位，第3位有且仅有1种选择，总数= $5*5*1*1*1=25$ 。

10. 一次期末考试，某班有15人数学得满分，有12人语文得满分，并且有4人语、数都是满分，那么这个班至少有一门得满分的同学有多少人？（ ）

A.23 B.21 C.20 D.22

答案：A

解析：容斥原理，总满分人数=数学满分+语文满分-语文数学满分=15+12-4=23。

11. 设A和B是两个长为n的有序数组，现在需要将A和B合并成一个排好序的数组，请问任何以元素比较作为基本运算的归并算法，在最坏情况下至少要做多少次比较？（ ）

A. n^2 B. $n\log n$ C. $2n$ D. $2n-1$

答案：D

解析：考虑2个数组分别是(1,3,5)和(2,4,6)，共需比较5次。因为结果数组大小是 $2n$ ，先从两数组取第一个值比较，小的入结果数组，剩下的和另一个数组的下一个数比较，依次这样，直到一个数组为空。另一个数组剩下的元素直接进结果数组。最坏一个数组空，另一个数组还剩1个元素，比较次数就是 $2n-1$ 。

12. 以下哪个结构可以用来存储图（ ）

A.栈 B.二叉树 C.队列 D.邻接矩阵

答案：D

解析：邻接矩阵和邻接表可以存储图，其他三项都是数据结构，不是存储结构。

13. 以下哪些算法不属于贪心算法？（ ）

A.Dijkstra算法 B.Floyd算法 C.Prim算法 D.Kruskal算法

答案：B

解析：Dijkstra算法需要每次选取 $d[i]$ 最小的边；Prim算法需要每次选在集合E中选取权值最小的边u；kruskal剩下的所有未选取的边中，找最小边。Floyd算法只需要按照顺序取边就可以了。

14. 有一个等比数列，共有奇数项，其中第一项和最后一项分别是2和118098，中间一项是486，请问一下哪个数是可能的公比？（ ）

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

答案：B

解析：设公比是 p ，那么 $2 \cdot p^{(2n-2)} = 118098$ ， $2 \cdot p^{(n-1)} = 486$ ，可以得到 $p^{(n-1)} = 243$ ，由于 $\gcd(2, 243) = \gcd(4, 243) = \gcd(5, 243) = 1$ ，所以排除2，4，5，而 $\gcd(3, 243) = 3$ ，所以公比可能是3。

15. 有正实数构成的数字三角形排列形式如图所示。第一行的数为 $a_{2,1}$ ， $a_{2,2}$ ，第n行的数为 $a_{n,1}$ ， $a_{n,2}$ ，...， $a_{n,n}$ 。从 $a_{1,1}$ 开始，每一行的数 $a_{i,j}$ 只有两条边可以分别通向下一行的两个数 $a_{i+1,j}$ 和 $a_{i+1,j+1}$ 。用动态规划算法找出一条从 $a_{1,1}$ 向下通道 $a_{n,1}$ ， $a_{n,2}$ ，...， $a_{n,n}$ 中某个数的路径，使得该路径上的数之和最大。

令 $C[i][j]$ 是从 $a_{1,1}$ 到 $a_{i,j}$ 的路径上的数的最大和，并且

$C[i][0] = C[0][j] = 0$ ，则 $C[i][j] = ()$

A. $\max\{C[i-1][j-1], C[i-1][j]\} + a_{i,j}$

B. $C[i-1][j-1] + C[i-1][j]$

C. $\max\{C[i-1][j-1], C[i-1][j]\} + 1$

D. $\max\{C[i][j-1], C[i-1][j]\} + a_{i,j}$

答案：A

解析：每个点的只能从 $C(i-1, j-1)$ 以及 $C(i-1, j)$ 过来，所以最优解肯定是从更大的那个节点到，所以结果包

含 $\max(C(i-1,j-1), C(i-1,j))$, 而计算的是和所以也包含 a_{ij} 这一项。

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

1.

```
1 #include <stdio>
2 using namespace std;
3 int n;
4 int a[100];
5
6 int main() {
7     scanf("%d", &n);
8     for(int i = 1; i <= n; ++i) {
9         scanf("%d", &a[i])
10     int ans = 1
11     for (int i = 1; i <= n; ++i) {
12         if ( i > 1 && a[i] < a[i-1])
13             ans = i ;
14         while (ans < n && a[i] >= a[ans+1])
15             ++ans;
16         printf("%d/n", ans);
17     }
18     return 0;
19 }
```

概述：12行if判断如 $a[i]$ 比前一位小，则从 i 开始，否则从上次开始14行while循环找ans向后找第一个 $>a[i]$ 的数12行的判断的意思是，如果后项 \leq 前项，则重新开始，否则从上项开始（蠕动）整个程序含义是找每个 $a[i]$ 后第一个大于 $a[i]$ 的位置（如果看懂，后面都很好做）

I 判断题

1) （1分）第16行输出ans时，ans的值一定大于i。（ ）

答案：错

解析：12行if成立，14行while不成立，则16行 $ans=i$

2) （1分）程序输出的ans小于等于n。（ ）

答案：对

解析：13行 $i \leq n$ ，15行 $ans < n$ 才会自增，所以不会超过n

3) 若将第12行的“<”改为“!=”，程序输出的结果不会改变。（ ）

答案：对

解析：改成!=，无非是多了一些无用的比较，最后结果不变其实12行直接删掉，结果也不会变，只是速度变慢而已

4) 当程序执行到第16行时，若 $ans-i>2$ ，则 $a[i+1] \leq a[i]$ 。（ ）

答案：对

解析：14行，由于ans是第一个大于 $a[i]$ 的，所以 $a[i+1]..a[ans-1]$ 都不超过 $a[i]$ ，结论成立

5) (3分) 若输入的a数组是一个严格单调递增的数列 , 此程序的时间复杂度是 () 。

- A. $O(\log n)$ B. $O(n^2)$ C. $O(n \log n)$ D. $O(n)$

答案 : D

解析 : 单调增 , 则12行if不会成立 , 也就是ans只增不减所以复杂度为 $O(n)$

6) 最坏情况下 , 此程序的时间复杂度是 () 。

- A. $O(n^2)$ B. $O(\log n)$ C. $O(n)$ D. $O(n \log n)$

答案 : A

解析 : 最坏情况下 , 12行if总是成立 (a单调降) 此时14行也会一直运行到ans=n , 复杂度为 $1+2+..+n=O(n^2)$

2.

```
1 #include<iostream>
2 using namespace std ;
3
4 const int maxn =1000;
5 int n;
6 int fa[maxn],cnt [maxn];
7
8 int getroot(int v ) {
9     if (fa[v] == v) return v;
10    return getroot(fa[v]);
11 }
12
13 int main ( ) {
14     cin >> n;
15     for (int i =0;i<n;++i){
16         fa[i]=i;
17         cnt[i]=1;
18     }
19     int ans = 0 ;
20     for (int i=0; i<n - 1; ++i){
21         int a,b,x,y;;
22         cin >>a>>b
23         x=getRoot(a);
24         y=getRoot(b);
25         ans +=cnt[x]*cnt[y];
26         fa[x]=y;
27         cnt[y] +=cnt[x];
28     }
29     cout<<ans<<endl;
30     return 0;
```

31 }

判断题

1) (1分) 输入的a和b值应在 $[0, n-1]$ 的范围内。 ()

答案：对

解析：从初始化看，下标范围为 $0 \sim n-1$ ，所以合并范围也在此内

2) (1分) 第16行改成“fa[i]=0;”，不影响程序运行结果。 ()

答案：错

解析：findRoot里用到fa[v]=v表示组长

3) 若输入的a和b值均在 $[0, n-1]$ 的范围内，则对于任意 $0 \leq i < n$ ，都有 $0 \leq fa[i] < n$ 。 ()

答案：对

解析：fa[i]表示i同组的上级，下标也在 $0 \sim n-1$ 范围内

4) 若输入的a和b值均在 $[0, n-1]$ 的范围内，则对于任意 $i < n$ ，都有 $1 \leq cnt[i] \leq n$ 。 ()

答案：对

解析：cnt表示子连通块大小

选择题

5) 当n等于50时，若a、b的值都在 $[0, 49]$ 的范围内，且在第25行时总是不等于y，那么输出为 ()

A. 1276 B. 1176 C. 1225 D. 1250

答案：C

解析：每两次合并x和y都不同，表示每次都是单独一个去和整体合并。此时cnt[y]增加cnt[x]的值，也就是加1。 $1*1+1*2+...+1*49=50*49/2=1225$

6) 此程序的时间复杂度是 ()

A. $O(n)$ B. $O(\log n)$ C. $O(n^2)$ D. $O(n \log n)$

答案：C

解析：并查集getRoot函数没有路径压缩，单次查找最坏为 $O(n)$ 。总效率为 $O(n^2)$

3. 本题t是s的子序列的意思是：从s中删去若干个字符，可以得到t；特别多，如果s=t，那么t也是s的子序列；空串是任何串的子序列。例如“acd”是“abcde”的子序列，“acd”是“acd”的子序列，但“acd”不是“abcde”的子序列。

S[x..y]表示s[x]...s[y]共y-x+1个字符构成的字符串，若x > y则s[x..y]是空串。t[x..y]同理。

```
1 #include <iostream>
2 #include <string>
3 using namespace std;
4 const int max1 = 202;
5 string s, t;
6 int pre[max1], suf[max1]
7
8 int main() {
```

```

9   cin >> s >> t ;
10  int slen =s. length(), tlen= t. length();
11  for (int l = 0 ,j = 0 ; i < slen; ++i) {
12    if (j < tlen&& s[i]==t[j] ) ++j;
13    pre[i] = j;// t[0..j-1]是s[0..i]的子序列
14  }
15  for (int l = slen -1 ,j= tlen -1; l >=0 ; --i) {
16    if(j >=0&& s[i] == t [j]) --j;
17    suf [i]= j; //t[j+1..tlen-1]是s[i..slen-1]的子序列
18  }
19  suf[slen] = tlen -1;
20  int ans = 0;
21  for ( int i=0, j=0, tmp=0; i <=slen; ++i ) {
22    while ( j <=slen && tmp >=suf[j] + 1 ) ++j ;
23    ans =max(ans, j - l - 1);
24    tmp = pre[i];
25  }
26  cout << ans << endl;
27  return 0;
28  }

```

提示：

t[0..pre[i]-1]是s[0..i]的子序列；

t[suf[i]+1..tlen-1]是s[i..slen-1]的子序列

判断题

1. (1分) 程序输出时，suf数组满足：对任意 $0 \leq i < \text{slen}$ ， $\text{suf}[i] \leq \text{suf}[i+1]$ 。()

答案：对

解析：suf[i]是满足t[suf[i]+1..tlen-1]为s[i..slen-1]子序列的最小值

那么t[suf[i+1]+1...tlen-1]是s[i+1..slen-1]的子序列=>t[suf[i+1]+1...tlen-1]也是s[i..slen-1]的子序列，但不是最小（最小值是suf[i]），因此suf[i+1]>=suf[i]，单独看15到19行程序也可以直接得出这个结论

2. (2分) 当t是s的子序列时，输出一定不为0。()

答案：错

解析：可以理解题目的输出：s中删去连续多少个字母后t仍然是s的子序列；或者直接用s=t='a'代入，结果是0

3. (2分) 程序运行到第23行时，“j-i-1”一定不小于0。()

答案：错

解析：第一轮执行22行时tmp=0,j=0不执行，因此这轮j-i-1就可能是负数

4 (2分) 当t是s的子序列时, pre数组和suf数组满足: 对任意 $0 \leq i < \text{slen}$, $\text{pre}[i] > \text{suf}[i+1]$.()

答案: 错

解析: 可以用简单的样例 (如 $t=s='a'$) 代入检验, 也可以根据pre和suf的定义: 如果t是s的子序列, 那么 $0 \sim \text{pre}[i]-1$, $\text{suf}[i+1]+1 \sim \text{lent}-1$ 这部分分别是 $s[0 \sim i]$, $s[i+1 \sim \text{lens}-1]$ 的子序列, 不会重叠, 所以有 $\text{pre}[i]-1 < \text{suf}[i+1]+1$, 也就是 $\text{pre}[i] \leq \text{suf}[i+1]+1$

选择题

5.若 $\text{tlen}=10$, 输出为0, 则 slen 最小为 ()

A. 10 B. 12 C. 0 D. 1

答案: D

解析: slen 是s的长度, 至少需要输入一个长度的字符串, 如果t不是s子序列那输出一定是0

6.若 $\text{tlen}=10$, 输出为2, 则 slen 最小为 ()

A. 0 B. 10 C. 12 D. 1

答案: C

解析: 输出是2说明s串删去两个连续元素后t仍是s的子序列, 因此删去后长度至少为10, 删前至少为12

三、完善程序 (单选题, 每题3分, 共计30分)

1 (匠人的自我修养) 一个匠人决定要学习n个新技术, 要想成功学习一个新技术, 他不仅要拥有一定的经验值, 而且还必须先学会若干个相关的技术。学会一个新技术之后, 他的经验值会增加一个对应的值。给定每个技术的学习条件和习得后获得的经验值, 给定他已有的经验值, 请问他最多能学会多少个新技术。

输入第一行有两个数, 分别为新技术个数n ($1 \leq n \leq 10^3$), 以及已有经验值 ($\leq 10^7$)。

接下来n行。第i行的两个整数, 分别表示学习第i个技术所需的最低经验值 ($\leq 10^7$), 以及学会第i个技术后可获得的经验值 ($\leq 10^4$)。

接下来n行。第i行的第一个数 m_i ($0 \leq m_i < n$), 表示第i个技术的相关技术数量。紧跟着 m_i 个两两不同的数, 表示第i个技术的相关技术编号, 输出最多能学会的新技术个数。

下面的程序已 $O(n^2)$ 的时间复杂完成这个问题, 试补全程序。

```
1 #include<stdio>
2 using namespace std;
3 const int maxn = 1001;
4
5 int n;
6 int cnt [maxn]
7 int child [maxn] [maxn];
8 int unlock[maxn];
9 int unlock[maxn];
10 int threshold [maxn],bonus[maxn];
```



```

11
12 bool find(){
13     int target=-1;
14     for (int i = 1;i < =n;++i)
15         if(①&&②){
16             target = i;
17             break;
18         }
19     if(target== -1)
20         return false;
21     unlock[target]=-1;
22     ③;
23     for (int i=0;i < cut[target];++i)
24         ④;
25     return true;
26 }
27
28 int main(){
29     scanf("%d%d",&n, &points);
30     for (int l =1; l < =n ; ++l = {
31         cnt [l]=0;
32         scanf("%d%d",&threshold[l],&bonus[l];
33     }
34     for (int i=1;i < =n;++i = {
35         int m;
36         scanf("%d",&m);
37         ⑤;
38         for (int j=0; j < m ; ++j = {
39             int fa;
40             scanf("%d", &fa);
41             child [fa][cnt[fa]]=i;
42             ++cnt[fa];
43         }
44     }
45     int ans = 0;
46     while(find())
47         ++ans;
48     printf("%d\n", ans);
49     return 0;
50 }

```

1) ①处应填 ()

- A. `unlock[i] <= 0`
- B. `unlock[i] >= 0`
- C. `unlock[i] == 0`
- D. `unlock[i] == -1`

答案：C

解析：unlock作用是看是否能解锁任务。根据对问题5的分析，在未解锁前它的值是还有几个依赖任务未解锁。那么解锁条件当然是0个依赖任务，因此是等于0

2) ②处应填 ()

- A. `threshold[i] > points`
- B. `threshold[i] >= points`
- C. `points > threshold[i]`
- D. `points >= threshold[i]`

答案：D

解析：很简单，解锁条件之二，经验点要大于等于任务的需求点

3) ③处应填 ()

- A. `target = -1`
- B. `- cnt[target]`
- C. `bbonus[target]`
- D. `points += bonus[target]`

答案：D

解析：经验点增加。A肯定不对，target后面还要用。B不对，因为cnt[i]是依赖i的任务。C也不对，bonus是只读的

4) ④处应填 ()

- A. `cnt [child[target][i]] -=1`
- B. `cnt [child[target][i]] =0`
- C. `unlock[child[target][i]] -= 1`
- D. `unlock[child[target][i]] =0`

答案：C

解析：从前面分析看出unlock是依赖的还没解锁的任务数，解锁一个任务，所有依赖这个任务的unlock值都要减1

5) ⑤处应填 ()

- A. `unlock[i] = cnt[i]`
- B. `unlock[i] =m`
- C. `unlock[i] = 0`
- D. `unlock[i] =-1`

答案：B

解析：m是任务依赖的任务数，从前面代码看出当unlock[i]为-1时表示解锁成功，那么D不对。A的话cnt[i]此时还没完成赋值，也不对。C有迷惑性，认为unlock是布尔值，但看题目m个依赖任务完成才能解锁该任务，所以不是单纯的布尔，需要每解锁一个前置任务就将unlock减1，直到为0

2. （取石子） Alice和Bob两个人在玩取石子游戏，他们制定了n条取石子的规则，第i条规则为：如果剩余的石子个数大于等于a[i]且大于等于b[i]，那么她们可以取走b[i]个石子。他们轮流取石子。如果轮到某个人取石子，而她们无法按照任何规则取走石子，那么他就输了，一开始石子有m个。请问先取石子的人是否有必胜的方法？

输入第一行有两个正整数，分别为规则个数n ($1 \leq n \leq 64$),以及石子个数m ($\leq 10^7$)。

接下来n行。第i行有两个正整数a[i]和b[i]。 ($1 \leq a[i] \leq 10^7, 1 \leq b[i] \leq 64$)

如果先取石子的人必胜，那么输出“Win”，否则输出“Loss”

提示：
可以使用动态规划解决这个问题。由于b[i]不超过，所以可以使用位无符号整数去压缩必要的状态。Status是胜负状态的二进制压缩，trans是状态转移的二进制压缩。

试补全程序。

代码说明:

“~”表示二进制补码运算符，它将每个二进制位的0变成1、1变为0；

而“^”表示二进制异或运算符，它将两个参与运算的数重的每个对应的二进制位一一进行比较，若两个二进制位相同，则运算结果的对应二进制位为0，反之为1。

U11标识符表示它前面的数字是unsigned long long 类型。

```
1 #include <cstdio>
2 #include<algorithm>
3 using namespace std ;
4
5 const int maxn =64;
6
7 int n,m;
8 int a[maxn],b[maxn];
9 unsigned long long status ,trans ;
10 bool win ;
11
12 int main(){
13     scanf("%d%d",&n,&m);
14     for (int i = 0; i<n;++i)
15         scanf("%d%d",&a[i],&b[i]);
```

```

16 for(int i =0;i < n;++i)
17     for(int j = i +L;j<n;++j)
18         if (aa[i]>a[j]){
19             swap(a[i],a[j])
20             swap(b[i],b[j])
21         }
22 Status = ①;
23 trans =0;
24 for(int i =1,j=0;i<=m;++i){
25     while (j<n && ②){
26         ③;
27         ++j;
28     }
29     win=④;
30     ⑤;
31 }
32 puts(win ? "Win" : "Loss" );
33 return 0;
34 }

```

解析：首先使用 $f(i)$ 表示有 i 个石子时，是否有必胜策略。所以 $f(i)=!f(i-b[j1]) \text{ or } !f(i-b[j2]) \dots$ ($a[j] \leq i$)，转换公式， $status$ 中每一位定义为 $win(i-j)$ ，也就是有 $i-j$ 有必胜策略。因此第一题初始状态为都必输，因为石子有0个，怎么样都是输的。然后 $trans$ 相当于记录当前状态下能够必胜的策略位置也就是 $b[j]$ 的集合，但是因为要注意这边 $trans$ 没有清0，因为我们考虑到事实上能转移的状态数是不会减少的，所以这边第二题选B，表示将当前的状态增加到 $trans$ 里面，同时第三题选择A表示的就是将 $b[j]$ 加到 $trans$ 里面（记录新增的能够必胜的位置），然后第4题相当于往 $status$ 记录新的必胜策略的位置（也就是 $trans$ ），所以按照上述的转移公式 $f()$ ，第4题答案也就是D了，因为先手必胜的情况等价于，当前状态下能走到的先手必输的情况不为空。最好将 $status$ 状态更新，具体就是将当前的 win 记录到 $status$ 的最低位上即可（第5题）

1) ①处应填 ()

A. 0 B. $\sim 0ull$ C. $\sim 0ull \wedge 1$ D. 1

答案：C

2) ②处应填 ()

A. $a[j] < i$ B. $a[j] == i$ C. $a[j] != i$ D. $a[j] > i$

答案：B

3) ③处应填 ()

A. $trans |= 1ull << (b[j] - 1)$
 B. $status |= 1ull << (b[j] - 1)$
 C. $status += 1ull << (b[j] - 1)$
 D. $trans += 1ull << (b[j] - 1)$

答案：A

4) ④处应填 ()

A. $\sim \text{status} \mid \text{trans}$

B. $\text{status} \ \& \ \text{trans}$

B. $\text{status} \mid \text{trans}$

D. $\sim \text{status} \ \& \ \text{trans}$

答案：D

5) ⑤处应填 ()

A. $\text{trans} = \text{status} \mid \text{trans} \wedge \text{win}$

B. $\text{status} = \text{trans} \gg 1 \wedge \text{win}$

C. $\text{trans} = \text{status} \wedge \text{trans} \mid \text{win}$

D. $\text{status} = \text{status} \ll 1 \wedge \text{win}$

答案：D