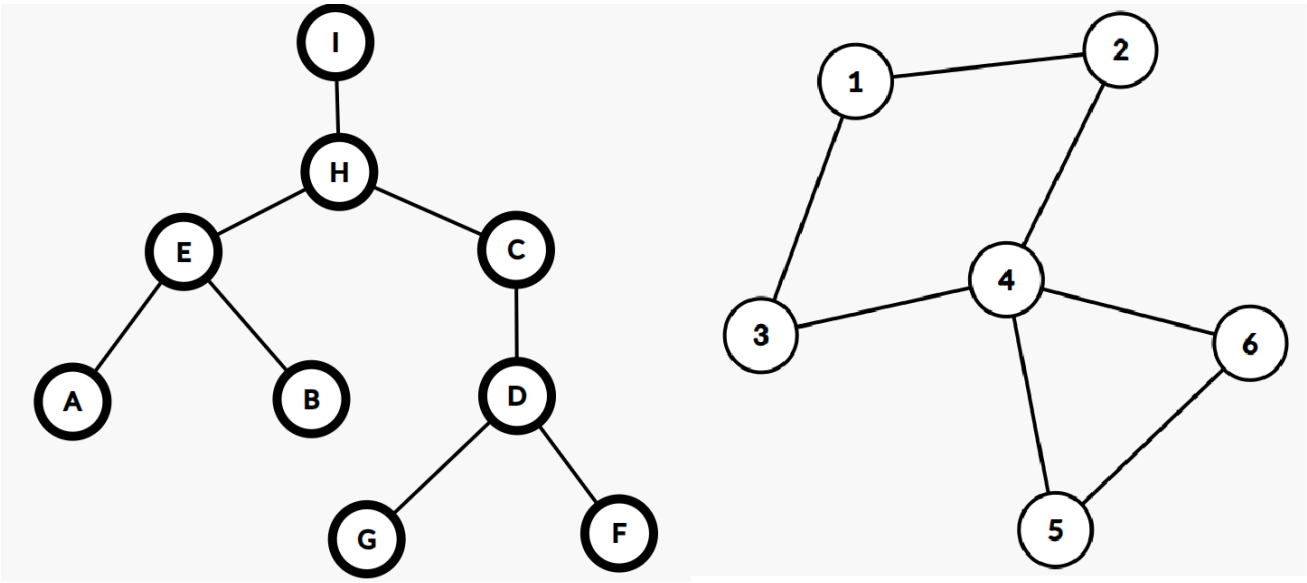
YDSP Junior 官方详解

快速对答案：CABDB CCDAD BAABC, TFTTDC, TFTFBCA, TFTTCB, CDAAC, BABDBCA

注：J 组第 15 题出错，A,C 都对。

1. 单项选择

1. 可以背，但是显然 A 和 B 是错误的，而 D 中 foundation 是基础，和中国计算机学会无关。
2. **注意集合在 J 组考纲内。** 化简得 $[1,144] \cup [199,211]$。
3. 简单的进制转换。$114_7 = 60$，$514_{16} = 1300$。
4. 对于这种栈的模拟题，我的推荐做法是拿出笔画一个栈，入栈就把字母写在最上面，出栈则把最顶上的字母划掉。
5. 转化出的中缀表达式应该是 `(x || y) && !(x && y)`。
6. D。哈夫曼编码也是每次选出频率最低的两个数进行编码。
7. 下方左图即为所求二叉树，其中 H, D 可以是父亲的左孩子或右孩子。



10. A,B 本身错误（B 考虑负数）；C 有溢出风险；D 分类讨论就可以证明。
11. 考虑插板法。但是注意到每个小朋友需要两个苹果，所以可以先每人分一个然后插板。剩下 21 个苹果有 20 个空隙，需要插两块板，第一块左边给第一个小朋友，中间给第二个，右边给第三个。所以，方案数就是 $\frac{20 \times 19}{2} = 190$。
12. 碰到区间和的问题，先在一张草稿纸上把四个区间画出来。观察发现，$S(1,12)$ 和 $S(7,12)$ 可以求 $S(1,6)$，而 $S(10,20)$ 和 $S(7,20)$ 可以求 $S(7,9)$，相加就是 $S(1,9)$ 了。
13. 阅读题。如果把 Pure 当成满分，那么 Far 的扣分就是 $\frac{5 \times 10^6}{n}$，而哪怕所有其他音符都是 Max Pure，也只够加 $(n-1)$ 分。因此，$n - 1 \geq \frac{5 \times 10^6}{n}$ 即可产生伪 PM，这需要 2237 个音符。

14. 模拟即可。坑点包括：
    1. `i` 从 1 开始，所以第一个 `y` 应该忽略。
    2. 判定条件是 `'A'<s[i] && s[i]<='Z'`，不包含 `A`。
    3. 如果大写字母出现在前半部分，那么被 swap 到后面之后会在 `s.size()-i` 处 swap 回来。
15. 一个较为细致的分类讨论题，图为上方右图。从 1 出发，有走 $2/3$ 两种选择，到达 4 后可以选择先 56 还是先 $3/2$；走 56 还可以讨论先走哪个，有 8 种选择。

    从 2 出发，如果走 $2-1-3-4$ 则有 2 种选择，走 $2-4$ 后有 4 种选择，共 6 种。

    从 4 出发，有 8 种选择：先走 123 还是先走 56，以及两个环的方向。

    从 5 出发，也是 6 种选择。

2. 阅读程序

第 1 题：浑身是坑的语法题

关于 `u.a[v.b[u.a][u.a]]`：注意到 `i[a]` 和 `a[i]` 都表示 `*(a+i)`，就有 `u.a[v.b[u.a][u.a]]` 等于 `u.a[u.a[v.b][u.a]]` 也就是 `u.a[u.a[u.a[v.b]]]` 即 `v.b` 了。

关于具体题目：

1. `sizeof` 的结果是一个 `unsigned long long`，所以 `sizeof sizeof v` 是 8。
2. 确实是 F。
3. `union` 的特征就是几个成员共用一块内存，现在 `a` 正在使用内存，所以输出 `b` 是 UB。
4. 考虑 `memset` 的结果，如果 `v.a[i]>0` 则至少为 `0x1010101`，所以 `v.a[i]>1:v.a[i]:1` 和 `v.a[i]>100:v.a[i]:1` 是一回事。
5. 不妨将这一整行代码展开得到 `cout<< 3<<1 * 3<<1 <<'\n'`，重新调整一下空格：`cout<< 3 << 1*3 << 1 << '\n';`，所以答案是 331。
6. 第 21 行 `case0:` 是没有空格的，所以输入 0 不会跳转到这里。

   `0110^101` 中，前者是八进制，后者是十进制，所以异或结果为 45，因此如果你输入 1，会先换行再输出 1，这个 1 会出现在第四行。

   有人会说 20202022020202 不可能出现，因为如果你输入 2 输出就是 2 ——然而你可以输入 258，转成 `unsigned char` 后还是 2。

第 2 题：星期计算程序

这个程序的功能还是比较好猜的，根据输入输出格式可以猜个八九不离十，关键在实现。

1. `check` 是闰年判断。那么在这一行中，假如 m 真的是 2，那么 `sum_m` 贡献只有一月的，和二月有几天没有关系。
2. 数组只开到了 3000，我们的预处理也仅到 3000，所以无法处理 4000 年。
3. 这一行是判断是否为 31 天。 要判断有没有影响，关键看 7 月和 8 月是否仍被认定为 `true`。
4. 不必要，哪怕 4000 年加起来总天数也远不及 $2^{31}-1$。
5. 核心在于第 Y 年的 12 月 31 日是否是星期 $base$。事实上 B,C 均正确，但 C 用到的 `cos` 函数所在的 `cmath` 头文件并未引入。
6. 略。

7. 逐个分析：
    - A 项错误，因为第 27 行使用到了 `d_m` 数组。
    - B 项正确，因为 `365 ^ 1` 与 364 等价，而 $364 \equiv 0 \pmod 7$。
    - C 项正确，因为没有关系。
    - D 项正确，因为对 ans 的两个计算式，顺序是无所谓的。

第 3 题：更相减损法的扩展

1. m 在之后的程序中没有被用到，且 m 在参数中没有 `&`，所以确实没有影响。
2. n, m 均总是在 $[1, 10^9]$ 范围内，不会溢出。
3. 注意到第 5 行处 n, m 都是最初 n, m 的最大公约数，而每次执行第 9 行时，此时的 n 一定也是最大公约数。这么看，最终的返回值就是下一层递归的值 + 最大公约数，也就是最大公约数乘以递归层数。所以交换 n, m，答案不变。
4. 确实，只要 $m = 0$，那么 `n-=m` 的结果就还是 n，然后就死循环了。
5. 不难想到只要 $m = 1$ 或 $n = 1$，另一个数就会疯狂减 1，所以时间复杂度是 $O(\max(n, m))$ 的。感性理解一下发现它和 $O(n + m)$ 相同。严格证明的话，$\frac{n+m}{2} \le \max(n, m) \le n + m$，所以忽略常数后它们确实是一回事。

3. 完善程序

第 1 题：双向链表上冒泡排序

程序分析：第 $8 \sim 19$ 行读入数据并倒序塞进链表。第 $20 \sim 21$ 行是首尾相接。

第 $22 \sim 35$ 行是核心部分，在 i, j 两个指针遍历的基础上，根据第 25 行我们知道第 26 到 33 行目的是交换 j 和 p 在链表中的顺序。

第 $36 \sim 40$ 行用于输出链表。

1. 存在 `tail->nxt`，排除 AD。观察加入元素的过程，全程没有 `tail`，所以如果 `tail` 是 `new node`，那么 `tail` 就永远接不上链表。所以 `tail` 就是 `head`。
2. 冒泡排序中，i 从前往后，则 j 一定从后往前。根据加数过程，`tail` 里面没有数字，所以 j 从 `tail->prv` 开始；最后到 `i->nxt` 结束，所以你要写 `j!=i`。
3. `Swap` 是交换 `prv` 和 `nxt`；`Symm` 是建立 `j,p` 的反向边；`Reve` 是交换 `j,p` 内部顺序。所以 `Reve` 可以放在任意位置，但是 `Symm` 必须在 `Swap` 之后。

    （P. S. 如果有的出题人没有设置中间变量 `p`，那么应该怎么写呢？）
4. 本空难点在于不知道这里要干啥。回顾数组中的冒泡排序，当交换 `a[i]` 和 `a[i+1]` 时，i 不会改变，但是在这里，因为 i 事实上是指向 a_i 的指针，所以交换后，i 也向后移动了。因此，我们要把这个“意外的”移动恢复，选 A。
5. 同样地，在交换时 i 会跟着跑，`head` 也会跟着跑，所以 `head` 早已不指向第一个元素。然而，`tail` 因没有元素而不参与交换，用上循环的性质，`tail->nxt` 就是第一个元素。

第 2 题：DFS

下文中，称“大行”为一些宫组成的行，“小行”是一些格子组成的行，列同理。

程序分析：`chkRow(x)` 判断第 x 大行是否合法，`chkColumn()` 判断全部列是否合法，这点和前面给出的思路一致。`rot(x,y)` 旋转一个宫。`dfs(c,r,s)` 搜索目前在第 c 列第 r 行，目前有 s 步的情况。

然后我们还需要知道各个数组在干什么。输入告诉我们 `base[i][j]` 存储的数独中的数字；然后观察第 $77\sim83$ 行，如果 s 更优秀就把 `res` 改成 `tmp`，所以 `tmp` 是搜索时每个位置旋转次数，`res` 是最优答案的旋转次数。

1. 其实你不需要完全弄明白 `rot` 里面的一堆 swap。注意到 `swap` 中用到了 $[x,x+3]$ 和 $[y,y+3]$，所以 n 应该等于 4。
2. 判定第 x 大行是否合法，就需要判定从第 $nx-n+1$ 小行到第 nx 小行是否合法，而 x 就是要变成 i 的下界。
3. 这两空的目的是判定是否存在重复数字，而 `buc` 自然就是 bucket（桶）的缩写。如果一个数曾经存在过，则有重复数字。
4. 这里问的是什么时候我们不应当接着搜。此时，一大行已经填完了，所以应当检查这一大行进行优化；但是大列可能没有填完，所以不能判定。
5. 这里问的是什么时候我们可以更新答案。那么显然我们需要已经填完了，并且列检查没问题才行。（行在每行末尾都检查过了）。
6. 这里是搜索的回溯，需要把“污染”了的数组全部还原。我们离开时只要恢复 `tmp` 即可。
7. 见“程序分析”。