当你打开这个小锦囊的时候，可能正处于被编程作业折磨的走投无路的状态。
如果是这样，请把脑子里乱糟糟的情绪放空，放空，放空，深呼吸。
当你处于一个相对祥和的状态时，再来看一下，我们的提示，能不能帮助你理清一些头绪。
如果它依然效果有限，也请不要绝望，带上你的截图和疑问，大胆私信我吧。

下面的提示分为初级提示、中级提示和高级提示。
初级提示一般仅仅提示基础思路，适合大脑一片空白的小朋友;
中级提示会提示到一些易错点，适合写出了程序框架但是觉得似乎有哪里不对的小朋友;
高级提示大概就很接近剧透了，适合差一点点就豁然开朗的小朋友。
三类提示会分别放在三页上，大家按需查看。

初级提示：

第 1 题：

基础思路：写一个 main 函数和一个 perfectNumber 函数。在 main 函数中让用户输入下界和上界，然后循环遍历该范围中的每个整数，每个周期中都调用 perfectNumber 函数来判断该整数是否为完全数。

第 2 题：

基础思路：在 myfun 的函数体中，运用指定的 for 循环对 7 个数进行累加和累乘，累乘时在不判断各个数是否为 0 的情况下做到把 0 变成 1 而不改变非零的数，同时在不判断各个数是否为 0 的情况下做到统计非零数的个数。最后在不判断奇偶的情况下做到根据非零数的个数奇偶性返回不同的值。

第 3 题：

基础思路：既然不能用循环，那就用递归。递归的核心是**“我调用我自己”**。本题里的“我”需要完成：判断当前的头尾字符是否相同；如果不同则**返回某个值**并结束；如果相同则判断是否满足终止条件，如果不满足则调用我自己后**返回某个值**。
在 main 函数调用 func 函数时，可以用 strlen 函数获取最初的字符串长度。

第 4 题：

基础思路：根据各个函数的不同需要，将角色状态、职业、技能或训练成功的概率等数组和所需的变量设计为函数的参数。对于只处理一位角色的数据且不包含角色序号的函数，应将二维数组中的一行或三维数组中的一个二维数组作为参数。

中级提示：

第 1 题：

易错点：
① 误把题目中表示上界的 n 和 perfectNumber(n)中的 n 当作同一个东西；
② 在 main 函数中不写循环，而在 perfectNumber 函数中遍历 m~n 之间的数；
③ 在判断是否为完全数时漏掉了 1，或多加了该数本身。

第 2 题：

易错点：
① 认为题目出错了，按题目要求不可能写出这个程序；
② 在限制要求 b、c 指定的循环体中写了条件判断、关系表达式或逻辑表达式，违反要求。

第 3 题：

易错点：递归调用自己时，忘了将回溯的返回值当做自己的返回值（想想为什么要这样做）。
注意：本题提供的 func 函数有两个参数，一是表示字符串起始地址的 array，二是表示字符串长度的 len。那么，“掐头去尾”也就意味着 array+1 和 len-2。

第 4 题：

易错点：
① 以常量为实参时，形参未加 const；
② 函数的形式参数是二维数组时，没有规定第二维的大小；
③ 漏看题目要求，在只处理一位角色的函数中使用角色序号或循环结构；
④ 重复设置随机数种子，导致为不同角色或不同天的训练产生相同的随机数。

高级提示：

第 1 题：

在 bool perfectNumber(int n)函数中循环遍历从 1 到 sqrt(n)+0.1 之间的整数，用%来判断该整数是否为 n 的约数。把所有约数都加起来，再判断是否与其本身相等，如果相等则返回 true，否则返回 false。

细节思考：为什么要遍历到 sqrt(n)+0.1，而不是 sqrt(n)?

第 2 题：

累乘时，不做判断怎么把 0 变成 1？提示：利用类型转换和算术表达式。
不做判断的情况下，如何选择要返回“和”还是“积”？提示：利用算术表达式。

第 3 题：

每次递归调用时，我们希望更新数组的起始地址和长度，而这两者刚好都是 func 函数的参数，所以只要让被调函数的 array（形参）获得主调函数的 array+1（实参）、被调函数的 len（形参）获得主调函数的 len-2（实参）即可。
因为 func 函数的最终目的是“判断是否为回文数”，所以它的返回类型应该是 bool。

第 4 题：

利用函数的返回值来统计训练成功的人次数。