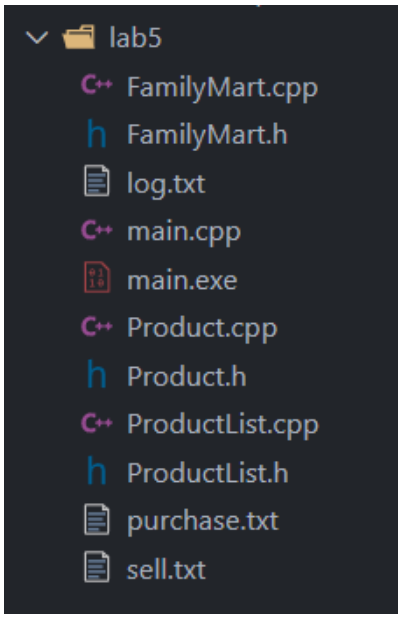
Lab5

一：问题回答

1：根据我的理解，重载是通过使多个同名函数的特征标（参数列表）不同来实现的，而默认参数是通过给一个函数的参数部分地赋默认值来实现的。也就是说，对于重载来说，参数列表不同的话即为不同的函数，函数内部的步骤可以不同；而对于默认参数来说，尽管在使用过程中可以提供不同的参数，其形式好像与重载很像，但带默认参数的函数内部的步骤都是一样的，只是在没有提供实参的地方使用了默认参数。

2：折扣只是一个数字，因此直接放入 ProductList 类的成员函数 SellProduct（）中了，然后这里 SellProduct（）函数要接受一个 double 类型的参数表示折扣。我这里是用了默认参数的做法，默认 discount=1.0，因为 SellProduct（）函数内的步骤都应该是一样的，不同的折扣只是一个量，用默认参数就够了，用重载的话显得很臃肿。

二：代码思路（请关注“件”（Product）、“种”（ProductList）字样）



1：如左图文件夹所示，首先是 Product 类，该类中包含了每**件**商品的各种信息；然后 ProductList 类是用一个链表把**同种**商品（name 相同）串起来，链表的节点为一个 Product 对象和一个 next 指针。ProductList 类中还有该**种**商品的数量的商品的名称信息；最后 FamilyMart 类也是用一个链表把所有商品**种**类串起来，链表节点为一个 ProductList 对象和一个 next 指针。上述便是商品信息存储的总体思路，具体可参考代码及注释。

2：进货操作的大致思路（请参考 FamilyMart：：purchase_file()函数的代码）

首先从文件中依次读取每**件**商品的各种信息并将其存储到临时变量中，然后在一**件**商品的信息读取完成后用这些临时变量生成一个临时商品对象 tempproduct。然后就是要存储该件商品的信息，首先在库存的商品**种**类中根据商品名称来寻找有没有该**种**商品，如果没有找到该商品**种**类的话则需要新创建一个商品**种**类然后把该**件**商品放入该商品**种**类中。如果找到的话则在该商品**种**类后添加该**件**商品，这里添加商品时我用的是插入排序的方法（请参考 ProductList：：AddProduct()函数的代码），由于在生成临时商品对象 tempproduct 时已经计算了商品的剩余保质期（请参考 Product：：LeftLife（）函数的代码），故这里根据剩余保质期从小到大进行插入排序，这样得到的每**种**商品 ProductList 中从头指针开始到尾指针每**件**商品的剩余保质期便是按从小到大排列的，这样在售卖商品和更新商品的时候只需从前往后直接进行操作即可，无需再进行一次遍历。

3：售货操作的大致思路（请参考 FamilyMart：：sell_fill()函数的代码）

首先从文件中依次读取要售卖的每**件**商品的名称和折扣，然后根据该**件**商品的名称从库存中寻找有没有该商品**种**类，如果没有找到则在命令行中提示“Sorry we don’t have “name” now”（已注释掉）,如果找到该**种**商品则检查该**种**商品的数量，如果数量为 0 则也进行提示“Sorry we don’t have “name” now” （已注释掉），如果数量不为 0 则进行售货并记录交易额。（请参考 ProductList：：SellProduct（）函数的代码）。

4：更新操作的大致思路（请参考 FamilyMart：：UpdateFamilyMart（）函数的代码）

注：该函数已整合到 FamilyMart::show_turnover()函数中。

首先要更新当前日期 NowDate【3】，之后依次遍历每**种**商品，再对每**种**商品依次遍历每**件**商品来更新库存中所有的商品的剩余保质期。对于每**种**商品清除剩余保质期<=0 的商品。（这里小于 0 属于运行出错，在命令行中会给出提示，正常的话剩余保质期应该最小为 0）

然后对于商品数量为 0 的商品**种**类则直接删除。(请参考 ProductList::UpdateProductList（）函数)。

5：保质期处理的大致思路：

在创建临时对象 tempproduct 时就根据当前日期 NowDate【3】来计算剩余保质期（请参考 Product:: LeftLife（）函数的代码)，如果进了已经过了保质期的商品则在进货时就直接忽略，然后在一天结束后更新剩余保质期使其减 1，然后再清除保质期为 0 的商品。

6：有关性能方面：

由于对于每**种**商品都有单独的一个链表（链表内包含该种商品下的一**件件**商品)，然后各**种**商品又组成一个链表，故在运行过程中内存是动态申请的，所占的内存取决于实际需要，即有多少**件**商品就申请多少**件**商品所需的内存。然后在运行方面，由于在存储上使用了商品名称索引，故进货时对于要储存的每件商品就只需遍历先商品**种**类，找到对应的商品种类（如没有则直接创建一个)，然后在该**种**商品下面再进行遍历，根据插入排序从小到大排列不同剩余保质期的一**件件**商品，之后售货时就只需根据名称索引遍历商品**种**类，确定该商品的数量然后直接弹出第一**件**商品（即头指针指向的商品，也是剩余保质期最短的商品）而无需在每**种**商品下再进行遍历。个人认为这种查字典式的遍历效率是比较高的。(鉴于最近课程压力，下面仅给出继续优化的想法，并未在本次 lab 中实现：①可以按照商品名称的首字符索引把商品名称再分类，然后每次只需查找字符和字符下的商品名称，在商品名称有很多时这样可以很大地提高效率。②然后可以考虑真实世界中新进的货物的剩余保质期大多大于库存货物，可以在同种商品下进行插入排序时从尾节点开始往前查找（本 lab 中是从头指针开始查找)，不过这样的话链表也要改成双向链表。)