Foods can be divided into those with significant fat (type F) and those that are virtually non-fat (type N). For this final you will be writing parts of a program that reads in a table of information about foods from about a file, food.txt, and then provides information about a food that a user enters on the keyboard. For this program you will have a base class Food, and a derived class FattyFood. You will note that only FattyFoods have information about their fat content. food.txt may have any number of entries in it. The program ends when the user enters "Done". You must use the STL, and may not use any arrays, including traditional C strings of chars. You will be using an overloaded >> operator and read() functions for input and display() functions for output. A STL map of strings and Food* holds the information. No data member of any class may be public.

food.txt format:

```
Food_type   Calories  Dietary_Fiber  Saturated_Fat (if a FattyFood)  Name
<char>      <int>     <int>          <float>                         <string>
F 290 2 4.0 Pizza
N 151 0 Cola
F 148 0 2.6 Popcorn
N 60 3 Orange
N 31 2 Carrot
```

Sample session:

```
lect2% food.out
Food (Done to end): Chips
Name not found.
Food (Done to end): Popcorn
148 calories, 0g fiber, 2.6g saturated fat
Food (Done to end): Carrot
31 calories, 2g fiber, and no fat.
Food (Done to end): Pizza
290 calories, 2g fiber, 4g saturated fat.
Food (Done to end): Done
lect2%
```

Common to Both Finals

The List class expects that T is a pointer to a class object. The list is sorted using the operator< of the T class. You should make use of that fact in your find() function(s). find() should return NULL if the item is not in the list.

```
template <class T>
class ListNode {
    T data;
    ListNode<T> *prev;
    ListNode<T> *next;
    ListNode(const T d, ListNode<T> *p, ListNode<T> *n);
    friend class List<T>;
};

template<class T>
class List {
    ListNode<T> *head;
    T find(const ListNode<T> *ptr, const T item);  // returns NULL if not found
public:
    List();
    T find(const T item);  // returns NULL if not found
    void insert(const T item);
};
```