CS2040C Data Structures & Algorithms Assignment #2

Objective

This assignment expands on the previous assignment. You will learn how to use C++ templates and operator overloads in order to make truly abstract data types that are flexible for use in any situation. There are two parts to the assignment. Part 1 applies templates to the linked list you implemented in assignment 1 so that any kind of data can be stored. In Part 2 you will use operator overloads in order to allow users to flexibly use your class.

Part 1: Template Linked List

You will be provided a zip file that contains:

- linked_list.h, a header file containing the definition of the template List class
- linked_list_test.cpp with a few public tests

Your task is to implement all of the methods of the List class marked as “TODO” in linked_list.h. You have been given a few starting test cases in linked_list_test.cpp, but satisfying those is not sufficient for full marks. You must code for all boundary conditions, which may not be covered in the basic tests. It is suggested that you write extra tests for any boundary conditions you can think of, but note that those are not submitted nor graded. You may modify any of the files, but will only submit the contents of linked_list.h. Note that you can add helper functions as needed, but cannot import other libraries. Specifically you must implement the following functions:

1. **contains(T)**: returns true if the linked list contains the item given.
2. **extract_max()**: finds the “largest” item in the list and returns that value. Also, delete the max element in the list afterwards.
3. **reverse()**: reverses the link list such that tail is now head and all items in between are in reverse order.
4. **to_string**(): returns a string representing the list and all items in it. For example, if head is 3, followed by 1 and then 4, to_string() should return “{3, 1, 4}”. Note that strings should be quoted, for example “{“a”, “b”, “c”}”. (Hint: Overloads a special to string for a single element, e.g. int, string.)

Submission

Submit to Coursemology the entire contents of linked_list.h. Coursemology will run a series of unit tests and provide you with feedback.

Part 2: Food

In the same zip file as Part 1, you will also find food.h and food.cpp, which define the Food class. The Food class contains a name and a number of calories. Your task is to implement the "+" operator so that you can write "foodA + foodB" and get a new food that has the two names concatenated with a space in between and the sum of their calories. For example, if "Chicken" has 300 calories and "Salad" has 100 calories, then Chicken + Salad produces "Chicken Salad" with 400 calories. You may modify any of the files and add helper functions as needed but only submit food.cpp.

Submission

Submit to Coursemology the contents of food.cpp **without** the `#include "food.h"` line. Coursemology will run a series of unit tests and provide you with feedback.

Extension Tasks

Here are some additional tasks that will further help you develop your C++ skills and prepare you for future assignments:

1. Implement the destructor, assignment, and copy constructor functions, marked "TODO: Optional" in linked_list.h
2. Implement the == and > operators for Food. Write unit tests for your new operators.
3. Can you simplify the existing unit tests using the new operators?
4. Write << (the stream operator) for both Food and List.
5. Write a new file that creates a menu as a List of Food items. Construct a menu and then print it.
6. Write an iterator for List and use it for the menu.