CS2040C Data Structure and Algorithm Lab Assignment #2

A zip file of the solution files for MS Visual Studio (or .zip of XCode file for Mac users) is provided that contains

- The Linked List class mentioned in lecture
    - `simpleLinkedListTemplate.h`
    - `simpleLinkedListTemplate.hpp` (or .cpp, it doesn't matter)
- A main file to use the Linked List: `main.cpp`
- And the files for class Food: `food.h`, `food.cpp`

You will **only** submit the modifications in these two files ONLY:

- `simpleLinkedListTemplate.h`
- `simpleLinkedListTemplate.cpp`

More precisely, you will only submit the following to coursemology only:

- The `List` class declaration (optional, only if you had modified it)
- `exist()`
- `extractMax()`
- `reverseOp()`
- And any new auxiliary functions you have added.

However, you can modify other files, especially for your own testing and debugging. But in our grading, we will assume all the other files are the not changed.

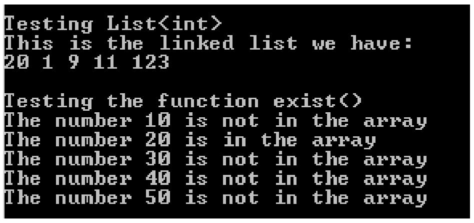
More precisely, you are only allowed to ADD more helping members or methods and you should NOT change the existing implemented declarations and implementations

Look into the `main.cpp` file. It contains the code:

```
int main() {
  testIntLL();
  testFoodExist();
  testFoodOpGreaterThan();
  testFoodAddition();
  testFoodSort();
  testReverseOp();
  return 0;
}
```

Task 1

The member function `exist()` is missing. Your job is to implement it in the same way you did in the previous assignment. Here is some sample output:



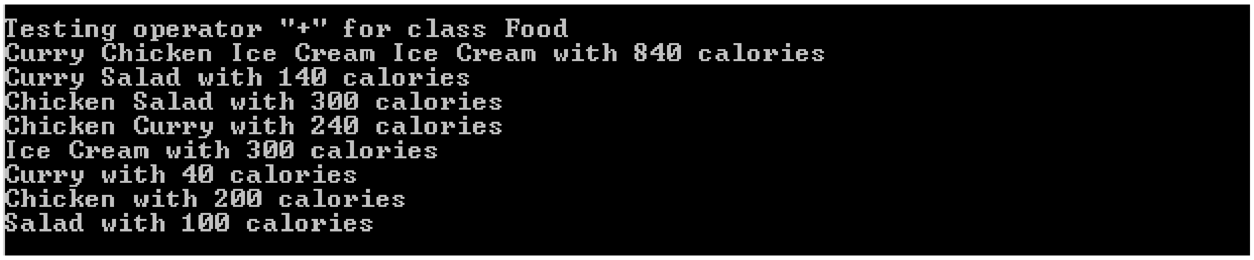
```
Testing List<int>
This is the linked list we have:
20 1 9 11 123

Testing the function exist()
The number 10 is not in the array
The number 20 is in the array
The number 30 is not in the array
The number 40 is not in the array
The number 50 is not in the array
```

Task 2

Task 2 is to implement the operation "+" for food. For example:

```
	Food food1("Salad", 100);
	Food food2("Chicken", 200);
	Food food3("Curry", 40);
	Food food23 = food2 + food3;
```

So the name of food23 will be Chicken Curry with 240 calories. Here is the sample output for the code given:



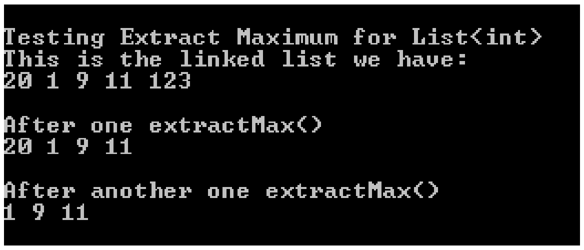
```
Testing operator "+" for class Food
Curry Chicken Ice Cream Ice Cream with 840 calories
Curry Salad with 140 calories
Chicken Salad with 300 calories
Chicken Curry with 240 calories
Ice Cream with 300 calories
Curry with 40 calories
Chicken with 200 calories
Salad with 100 calories
```

Task 3

To implement the function `extractMax()` in `List<T>.` This function will

- Find the maximum item in the list
- delete the node in the list and return the maximum item

Sample output:

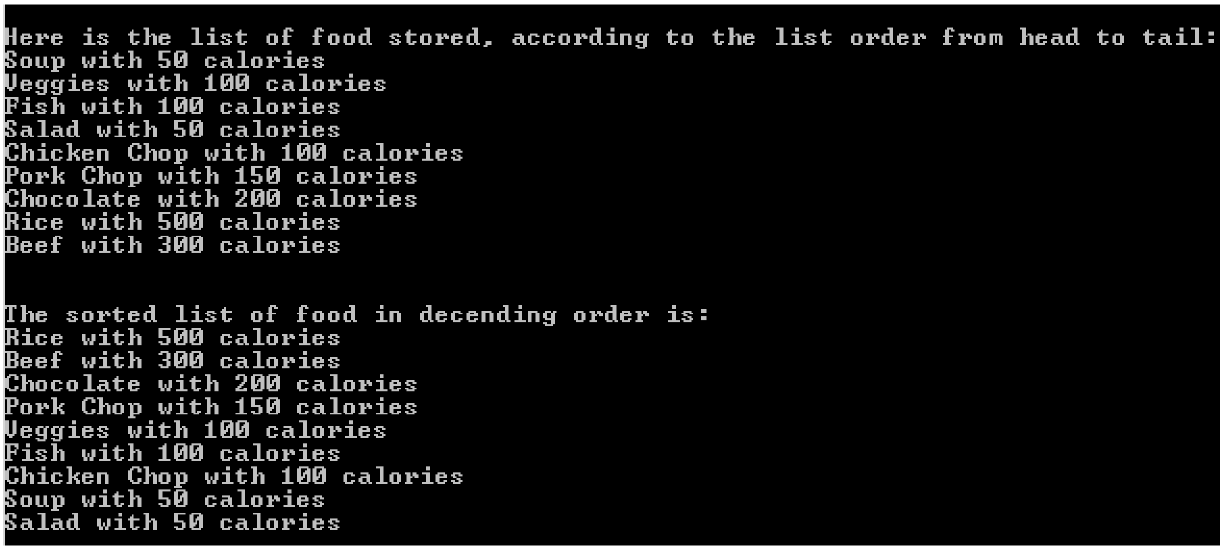


```
Testing Extract Maximum for List<int>
This is the linked list we have:
20 1 9 11 123

After one extractMax()
20 1 9 11

After another one extractMax()
1 9 11
```

If you have done it right, you can use it to sort the list by:

```
while (!l.empty())
	cout << l.extractMax() << " " ;
cout << endl
```

And it should work for any class like Food also, if you implement it correctly:



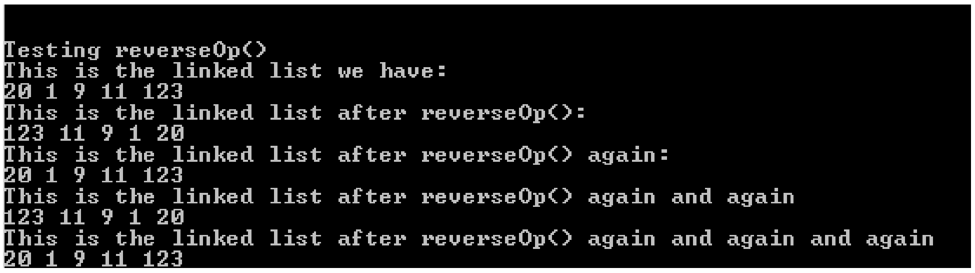
```
Here is the list of food stored, according to the list order from head to tail:
Soup with 50 calories
Veggies with 100 calories
Fish with 100 calories
Salad with 50 calories
Chicken Chop with 100 calories
Pork Chop with 150 calories
Chocolate with 200 calories
Rice with 500 calories
Beef with 300 calories


The sorted list of food in decending order is:
Rice with 500 calories
Beef with 300 calories
Chocolate with 200 calories
Pork Chop with 150 calories
Veggies with 100 calories
Fish with 100 calories
Chicken Chop with 100 calories
Soup with 50 calories
Salad with 50 calories
```

Task 4

Write down your thoughts. What are the disadvantages of the "LinkedList extractMax Sorting"? List at least two disadvantages in coursemology.

Task 5

Implement the function `reverseOp()` in the class `List`. It will reverse the order of the items in the list. Each time you call `l.reverseOp()` to modify "itself". Sample output from the function `testReverseOp()`:



```
Testing reverseOp()
This is the linked list we have:
20 1 9 11 123
This is the linked list after reverseOp():
123 11 9 1 20
This is the linked list after reverseOp() again:
20 1 9 11 123
This is the linked list after reverseOp() again and again
123 11 9 1 20
This is the linked list after reverseOp() again and again and again
20 1 9 11 123
```

Extra Tasks (Not Graded)

If you are looking for extra challenges, you could try the followings by yourself.

- Implement the "==" operators for the class Food
- Google how did we do "`cout << food;`"
- Implement another class to use it with the `LinkedList` Template. Look at the slides for an example of a class called "`Hero`".