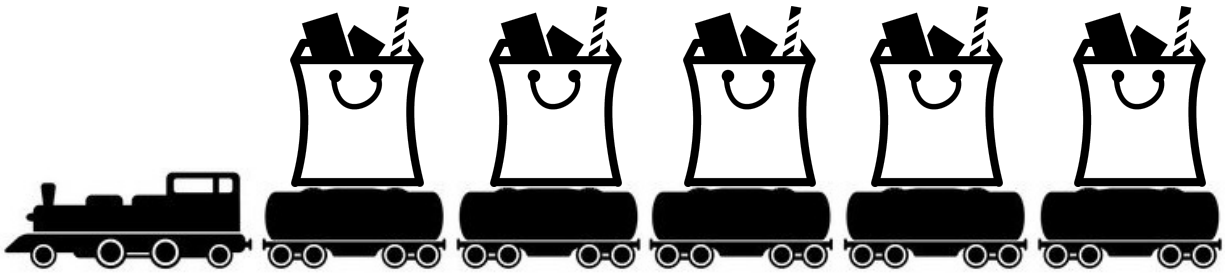
Project 2A: Another Bag



The specification of this project will feel very familiar, in that it is almost the same as that of project 1C. You will add union, intersection (without duplicates) and difference operations to the LinkedBag class, and a few more. You have thought about how to implement these functions already, so now you can focus on getting accustomed to manipulating nodes and linked chains.

We will work here with the `LinkedBag` class discussed during lecture. You will find the `Node` and `LinkedBag` class on Blackboard under Course Materials / Project2.

First you **must read the `LinkedBag` interface** and understand how it works. You will need to know how to use `LinkedBag` objects by understanding its interface.

Note: exam questions will be directly based on projects and ADTs/classes discussed in lecture. Reading and understanding the LinkedBag class will be fundamental to your success on the exams as well as on the project.

Implementation:

You will modify the `LinkedBag` class to add the following **public** member functions (note that you can implement some of these functions by calling other `LinkedBag` member functions):

```
/**
 @param a_bag to be combined with the contents of this (the calling) bag
 @return a new LinkedBag that contains all elements from this
 bag (items_)and from a_bag. Note that it may contain duplicates
 */
LinkedBag<T> bagUnion(const LinkedBag<T>& a_bag) const;
```

Note: this time bag size is not statically constrained as with arrays, so the union will contain all contents of both bags.

```
/**
 @param a_bag to be intersected with the contents of this (the calling)
 bag
 @return a new LinkedBag that contains the intersection of the contents
 of this bag and those of the argument a_bag. This intersection does not
 contain duplicates (e.g. every element occurring in BOTH bags will be
 found only once in the intersection, no matter how many occurrences in
 the original bags) as in set intersection A ∩ B
 */
LinkedBag<T> bagIntersectionNoDuplicates(const LinkedBag<T>& a_bag)
 const;
```

```
/**
 @param a_bag to be subtracted from this bag
 @return a new LinkedBag that contains only those items that occur in
 this bag or in a_bag but not in both, and it does not contain duplicates
 */
LinkedBag<T> bagDifference(const LinkedBag<T>& a_bag) const;
```

```
 /**
 @param a_bag whose contents are to be copied to this (the calling) bag
 @post this (the calling) bag has same contents as a_bag
 */
 void operator= (const LinkedBag<T>& a_bag);
```

```
/**
 @param new_entry to be added to the bag
 @post new_entry is added at the end of the chain preserving the order of
 items in the bag
 @return true if added successfully, false otherwise
 */
bool addToEnd(const T& new_entry);
```

Extra Credit:

```
/**
 @param an_entry to be removed from the bag
 @post the first occurrence of an_entry starting from the head node is
 removed from the chain preserving the order of the remaining items in
 the bag
 @return true if removed successfully, false otherwise
 */
bool removeRetainOrder(const T& an_entry);
```

IMPORTANT: Please remember that you DO NOT compile (or include in your project) the implementation (.cpp) of a Template class. Please look at slide 38 from Lecture 3 and make sure you understand separate compilation with templates (it will probably help if, as suggested on slide 40, you first run a dummy test and make sure you can compile a simple/trivial template class

Testing:

Gradescope will generate random bags of integers and test your functions for correctness. You should do the same (you may use any type you want on which the == operator is defined)

Write a main function that instantiates LinkedBag objects and calls the functions you implemented with different test values. Make sure to include edge cases (first item, last item, empty bag etc).

Submission:

For both regular submission and extra credit you must submit your implementation of LinkedBag only - **1 file: LinkedBag.cpp**

Your project must be submitted on Gradescope. Submission will open 48 hours before the due date. The due date is Friday March 3 by 6pm. No late submissions will be accepted.

Have Fun!!!!!