CS3305 Lab 6

Objectives

The purpose of this lab is to reinforce dynamic container class concepts and linked lists in C++. The labs consists of following problems:

Requirements

1. A Different dynamic `bag` (50 points)

   For dynamic `bag`, implement the operators `-=` and `-` for the class.

   - These operators compute the difference between two `bags`. In summary `b1-b2` results in a bag that has the elements of `b1` with the elements of `b2` removed.
     - For example, if `b1` has eight 4's in it and `b2` has five 4's in it then `b1-b2` has three 4's in it.
   - If `b2` has more of an element than `b1`, it is not an error. The difference simply ends up with none of that element.
     - For example, if `b1` has five 4's in it and `b2` has eight 4's in it then `b1-b2` has no 4's in it.
   - Implement `-=` as a member function and `-` as a non-member function (similar to += and +).
   - Here are the two headers from `bag.h`:
     - `void operator -=(const bag &subtrahend);`
     - `bag operator-(const bag & b1, const bag & b2);`
   - The program `bag_diff3.cpp` provide a basic check of the two new operators.

2. Get familiar with the methods in the linked lists toolkit (50 points)

- Create a project for this lab
  - Get the files `node1.h` and `node1.cpp` that contain the linked list toolkit and add them to the project
  - Include the files `check_lists.cpp` and `check_lists.h` in the project.
  - Create a C++ file that will contain a `main` program.
- Create a function that will print a list out, given the header node. Here is the function header:

`void list_print(node * head_ptr)`

   - Place this function before the `main` function in your program.
   - The function should print out the elements in the list separated by spaces all on one line
   - The function should print an end of line after the list data is printed.
- Carry out the following steps in the program, that is, in the function `main`
   - Create a list header and insert the following data in order: 23.5, 45.6, 67.7, 89.8, 12.9
      - Print out the list using the function you wrote
      - Call the function `check_list1` with the list as argument. This will print a single message if successful. Otherwise the program will terminate.
   - Create a list with two pointers, one to the head and one to the tail.
      - Insert 23.5 into the list
      - Then insert these elements in order at the tail of the list: 45.6, 67.7, -123.5, 89.8 and 12.9
      - Print the list
      - Call the function `check_list2` with the list as argument
   - Declare head and tail pointers for another list
      - Use those pointers to make a copy of the first list you created
      - Print the list
      - Call the function `check_list1` with the list as argument.
      - Print out the data at the tail of the list, it should be 23.5
   - Remove the first item in the second list created above
      - Print the list
      - Call the function `check_list2B` with the list as argument
   - Continue with the same list and remove the third item in the list
      - Print the list
      - Call the function `check_list2C` with the list as argument.
- Here is the output from one version of the exercise. This uses a fancier version of the print list function

```
{12.9, 89.8, 67.7, 45.6, 23.5}
check_list1 done
{23.5, 45.6, 67.7, -123.5, 89.8, 12.9}
check_list2 done
at location 4 in list2 -123.5
{12.9, 89.8, 67.7, 45.6, 23.5}
check_list1 done
at tail3: 23.5
{45.6, 67.7, -123.5, 89.8, 12.9}
check_list2B done
{45.6, 67.7, 89.8, 12.9}
check_list2C done
```

Submit your work using the appropriate link in D2L.