CS 162 Computer Science II

Assessment 3 Practice (Linked Lists I)

Academic Integrity

You may NOT, under any circumstances, begin an assessment by looking for completed code on StackOverflow or Chegg or any such website, which you can claim as your own. Please check out the Student Code of Conduct at PCC.

This assessment is individual work only. You must not discuss this assessment with any person other than your instructor. You may use course material for reference, but you must not use any other (non-course) material during this assessment.

I may ask you to explain your assessment verbally. If you cannot satisfactorily explain what your code does, and answer questions about why you wrote it in a particular way, then you should expect a zero.

This practice assessment focuses on the implementation of Linked Lists Part I.

This is a timed assessment, individual work only. Please do not discuss this assessment with any person other than the instructor.

Please be sure to submit something before the deadline! Late submissions are not accepted, and will receive a zero score. Submitting something, even if it does not work as well as you would like, is always better than not submitting anything and receiving a zero score!

Tasks

Start with the code in `~lliang/cs162/assessments/topic3` - You will want to copy it into a directory of your own:

```
cp -r ~lliang/cs162/assessments/topic3 your_directory
```

Use the provided makefile to compile your code:

- To compile: `make`
- To run: `./main`

- To clean up: `make clean`

Your output should look like this:

```
Current Inventory:
apple    0.99   Sun 2022-03-20
banana   0.69   Wed 2021-04-28
cookie   0.50   Sun 2023-02-12
donut    1.00   Thu 2021-10-28
egg      3.88   Sun 2022-01-30
fish     5.88   Mon 2021-02-15
milk     2.99   Tue 2021-03-02
yogurt   6.38   Sun 2021-06-13
```

You can modify the provided source files, header files and `makefile` as needed. `items.txt` contains a set of sample data. It's used by the program to populate the inventory. You are not allowed to use <string>, <vector> or any other header files in STL.

Check out the Sample Runs at the end of this section

You will find topic 3's code is quite similar with the code in topic 2. The only changes happened in the `ItemList` module, `itemList.h` and `itemList.cpp`. Here are some key differences:

1. The implementation for the `ItemList class` is changed from dynamically allocated array to a linear linked list (or singular linked list).
2. A `struct Node` is added to the private section of the `ItemList class`. This prevents the code outside of the `ItemList class` from accessing the `Node`, but it provides direct access to the member functions in the class.
3. Three private helper functions are added: `destroy, copy` and `append`. They are used by the public functions of the class.
4. You should get yourself familiar with two traverse schemes:
    a. One pointer traverse, e.g. use `curr` only. Usually, we use this scheme when we only visit the list
    b. Two pointers traverse, e.g use `prev` and `curr`. Usually, we use this scheme when we need to modify the list. It's much more convenient when you have a pointer to the previous node when you are trying to insert/remove.

For this test, you will need to complete the following tasks.

1. Create a public member function `getLowerCostItems()` for `class ItemList` to return the items that cost less than the passed in "`maxPrice`" in the list. The items are returned through the dynamically allocated array "`lowerCostItems`" and the function returns the number of items found that cost less than "`maxPrice`."
   `int getLowerCostItems(float maxPrice, InventoryItem *& lowerCostItems) const;`

First add the member function prototype inside `class ItemList` in `itemList.h`, then put the member function implementation in `itemList.cpp` and finally invoke/test the function in `main`.

You should read in the `maxPrice` from the user. Remember to prompt the user! You need to display the returned lower cost items. Please label your output clearly.

Hint: Remember to allocate the array before you try to populate it and also remember to deallocate the array after you are done using it. If you forget to allocate the array, your code will crash. If you forget to deallocate the array, you will get memory leaks. Many people hate C++ for these two frequently encountered enemies, segmentation fault and memory leaks.

2. Swap the data items in the first node and the last node. If there are less than 2 nodes, the function does nothing. Output the list after the swap and label it clearly.

   ```
   void swapFirstAndLast();
   ```

Use the provided makefile to compile your code:

- To compile: `make`
- To run: `main`
  Note: If you don't have '.' In your PATH environment variable, you need to type: `./main`
- To check memory leaks: `valgrind --leak-check=full ./main`
- To clean up: `make clean`

After you write your functions your Sample Run could look like this:

```
Current Inventory:
apple    0.99   Sun 2022-03-20
banana   0.69   Wed 2021-04-28
cookie   0.50   Sun 2023-02-12
donut    1.00   Thu 2021-10-28
egg      3.88   Sun 2022-01-30
fish     5.88   Mon 2021-02-15
milk     2.99   Tue 2021-03-02
yogurt   6.38   Sun 2021-06-13

Enter the maximum price and I will print the items below that price $: 5.00

There are 6 items below that price.
The items in your list below that price are:
apple    0.99   Sun 2022-03-20
banana   0.69   Wed 2021-04-28
cookie   0.50   Sun 2023-02-12
donut    1.00   Thu 2021-10-28
```

```
egg       3.88   Sun 2022-01-30
milk      2.99   Tue 2021-03-02

After swapping the first and last item the list is:
yogurt    6.38   Sun 2021-06-13
banana    0.69   Wed 2021-04-28
cookie    0.50   Sun 2023-02-12
donut     1.00   Thu 2021-10-28
egg       3.88   Sun 2022-01-30
fish      5.88   Mon 2021-02-15
milk      2.99   Tue 2021-03-02
apple     0.99   Sun 2022-03-20
```

Testing and Output on the Linux server

- ❏ Record 3 runs of your test by following the instructions below. Watch this video to see how to use the script command.
    - ❏ Type "`script output.txt`" on the command line and it will start recording your session in a file called "`output.txt.`"
    - ❏ Run your program as usual and test it with some sample input. Do this 3 times with different inputs.
    - ❏ You are required to use `valgrind` to check memory leaks when recording the 3 runs.
    - ❏ Type "`exit`" to stop recording.
    - ❏ The file should contain three runs of your test. Open `output.txt` and make sure it has the three runs of your test.

Submission

1. Copy the function(s) and the test code (how you invoke the functions) you have written to solution.txt
2. Transfer your `solution.txt` and `output.txt` from the PCC Linux server and upload to the D2L assignment. For more information on transferring files, please review Chapter 2 in the PCC Linux and vim Manual and watch the Linux Tutorial Videos in the D2L shell.
3. Do your own work. Consult the syllabus for more information about academic integrity. First add the member function prototype inside `class ItemList` in `itemList.h`, then put the member function implementation in `itemList.cpp` and finally invoke/test the function in `main`.

You should display the list after the swap. Please label your output clearly.

Use the provided makefile to compile your code:

- To compile: `make`
- To run: `main`
  Note: If you don't have '.' In your PATH environment variable, you need to type: `./main`
- To check memory leaks: `valgrind –leak-check=full ./main`
- To clean up: `make clean`