ECS 36C: Programming Assignment #2

Instructor: Aaron Kaloti

Winter Quarter 2022

Contents

1 Changelog

You should always refer to the latest version of this document.

- v.1: Initial version.

2 General Submission Details

Partnering on this assignment is prohibited. If you have not already, you should read the section on academic misconduct in the syllabus.

This assignment is due the night of Sunday, 01/30. Gradescope will say 12:30 AM on Monday, 01/31, due to the "grace period" (as described in the syllabus). *Be careful about relying on the grace period for extra time; this could be risky.*

3 Purpose of This Assignment

- To make you familiar with a linear data structure that is similar (but not identical) to a linked list.

4 Reference Environment

The autograder, which is talked about at the end of this document, will compile and run your code in a Linux environment. That means that you should make sure your code compiles and behaves properly in a sufficiently similar environment. The CSIF, which I talk about in the syllabus, is one such environment. Do not assume that because your code compiles on an *insufficiently* similar environment (e.g. directly on your Mac laptop), it will compile on the Linux environment used by the CSIF.

You should avoid causes of undefined behavior in your code, such as uninitialized variables. If you have things like this in your code, then your code could end up generating the wrong answer when autograded, even if it generated the right answer when you tested it in a sufficiently similar environment.

---

5   Cursor List

In this assignment, you will implement a cursor list. I talked about cursor lists during the 01/21 lecture.

Your code for this assignment must be written in C++.

In the `cursor_list.cpp` file provided to you on Canvas, the `NUM_SLOTS` macro, `struct Slot`, and the `SLOTS` array are defined. **You are NOT allowed to modify the definitions of these.** For instance, you cannot add a member to `struct Slot`.

You are NOT allowed to use any of the following:

- Pointers.
- Dynamic memory allocation.
- Global variables, except:
  - The `SLOTS` array provided to you.
  - You can add *one* global variable of type `bool`.
- Any STL container, such as `std::vector`.

Compliance with the above restrictions will be checked manually, and violating any of them may result in a heavy penalty, *perhaps even an automatic score of zero points.*

The *only* file that you will submit for this assignment is `cursor_list.cpp`. On Canvas, you are provided with `cursor_list.hpp` and a skeleton version of `cursor_list.cpp` meant to help you get started. Since the autograder will ignore your `cursor_list.hpp` file if you submit it, you are effectively *not allowed* to modify this header file.

In this assignment, you will implement a cursor list by implementing all of the functions that are declared in `cursor_list.hpp`. Each function declaration in `cursor_list.hpp` is preceded by a block comment explaining what the function does.

In cases in which an exception needs to be thrown, the autograder will not check the exact message that you use, but the message should be sensible.

Below are examples of how your code should behave. You can find the demo files on Canvas. **In the output, there are some parts that your version need not match exactly.**

```cpp
$ cat demo_cursor_list.cpp
#include "cursor_list.hpp"

#include <iostream>

int main()
{
    int l1 = clCreate();
    int l2 = clCreate();
    int l3 = clCreate();
    std::cout << clLength(l3) << ' ' << clIsEmpty(l3) << '\n';  // 0 1
    printSlots(0, 7);
    std::cout << "==========\n";
    clAppend(l1, 10);
    clAppend(l2, 20);
    clAppend(l3, 30);
    clAppend(l1, 40);
    std::cout << clLength(l1) << ' ' << clIsEmpty(l1) << '\n';  // 2 0
    std::cout << clFind(l1, 30) << '\n';  // 0
    std::cout << clFind(l1, 40) << '\n';  // 7
    printSlots(0, 10);
    std::cout << "**********\n";
    clInsert(clFind(l1, 10), 50);
    clInsert(clFind(l1, 10), 60);
    clPrint(l1);  // 10 60 50 40
    std::cout << "$$$$$$$$$$\n";
    clDelete(l1, 50);
    clPrint(l1);  // 10 60 40
    std::cout << clLength(l1) << '\n';  // 3
    std::cout << "@@@@@@@@@@\n";
    int l4 = clCopy(l1);
    clPrint(l4);
    std::cout << "&&&&&&&&&&\n";
    printSlots(0, 30);
    std::cout << "##########\n";
    std::cout << "Free list:\n";
    clPrint(0);
    clDestroy(l1);
    printSlots(0, 15);
    std::cout << "~~~~~~~~~~\n";
    std::cout << "Free list: (data doesn't matter)\n";
```

2

```
42        clPrint(0);
43        std::cout << "!!!!!!!!!!!!\n";
44        clAppend(14, 70);
45        clInsert(14, 80);   // insert at front
46        std::cout << "Before reverse:\n";
47        clPrint(14);   // 80 10 60 40 70
48        std::cout << "After reverse:\n";
49        clReverse(14);   // 70 40 60 10 80
50        clPrint(14);
51   }
52   $ g++ -Wall -Werror -std=c++14  demo_cursor_list.cpp cursor_list.cpp -o demo_cursor_list
53   $ ./demo_cursor_list
54   0 1
55   SLOTS[0]: 0 4
56   SLOTS[1]: 0 0
57   SLOTS[2]: 0 0
58   SLOTS[3]: 0 0
59   SLOTS[4]: 0 5
60   SLOTS[5]: 0 6
61   SLOTS[6]: 0 7
62   ==========
63   2 0
64   0
65   7
66   SLOTS[0]: 0 8
67   SLOTS[1]: 0 4
68   SLOTS[2]: 0 5
69   SLOTS[3]: 0 6
70   SLOTS[4]: 10 7
71   SLOTS[5]: 20 0
72   SLOTS[6]: 30 0
73   SLOTS[7]: 40 0
74   SLOTS[8]: 0 9
75   SLOTS[9]: 0 10
76   **********
77   SLOTS[1]: (header)
78   SLOTS[4]: 10
79   SLOTS[9]: 60
80   SLOTS[8]: 50
81   SLOTS[7]: 40
82   $$$$$$$$$$
83   SLOTS[1]: (header)
84   SLOTS[4]: 10
85   SLOTS[9]: 60
86   SLOTS[7]: 40
87   3
88   @@@@@@@@@@
89   SLOTS[8]: (header)
90   SLOTS[10]: 10
91   SLOTS[11]: 60
92   SLOTS[12]: 40
93   &&&&&&&&&&
94   SLOTS[0]: 0 13
95   SLOTS[1]: 0 4
96   SLOTS[2]: 0 5
97   SLOTS[3]: 0 6
98   SLOTS[4]: 10 9
99   SLOTS[5]: 20 0
100  SLOTS[6]: 30 0
101  SLOTS[7]: 40 0
102  SLOTS[8]: 50 10
103  SLOTS[9]: 60 7
104  SLOTS[10]: 10 11
105  SLOTS[11]: 60 12
106  SLOTS[12]: 40 0
107  SLOTS[13]: 0 14
108  SLOTS[14]: 0 15
109  SLOTS[15]: 0 16
110  SLOTS[16]: 0 17
111  SLOTS[17]: 0 18
112  SLOTS[18]: 0 19
113  SLOTS[19]: 0 20
114  SLOTS[20]: 0 21
115  SLOTS[21]: 0 22
```

3

```
SLOTS[22]: 0 23
SLOTS[23]: 0 24
SLOTS[24]: 0 25
SLOTS[25]: 0 26
SLOTS[26]: 0 27
SLOTS[27]: 0 28
SLOTS[28]: 0 29
SLOTS[29]: 0 0
##########
Free list:
SLOTS[0]: (header)
SLOTS[13]: 0
SLOTS[14]: 0
SLOTS[15]: 0
SLOTS[16]: 0
SLOTS[17]: 0
SLOTS[18]: 0
SLOTS[19]: 0
SLOTS[20]: 0
SLOTS[21]: 0
SLOTS[22]: 0
SLOTS[23]: 0
SLOTS[24]: 0
SLOTS[25]: 0
SLOTS[26]: 0
SLOTS[27]: 0
SLOTS[28]: 0
SLOTS[29]: 0
SLOTS[0]: 0 7
SLOTS[1]: 0 13
SLOTS[2]: 0 5
SLOTS[3]: 0 6
SLOTS[4]: 10 1
SLOTS[5]: 20 0
SLOTS[6]: 30 0
SLOTS[7]: 40 9
SLOTS[8]: 50 10
SLOTS[9]: 60 4
SLOTS[10]: 10 11
SLOTS[11]: 60 12
SLOTS[12]: 40 0
SLOTS[13]: 0 14
SLOTS[14]: 0 15
~~~~~~~~~~
Free list: (data doesn't matter)
SLOTS[0]: (header)
SLOTS[7]: 40
SLOTS[9]: 60
SLOTS[4]: 10
SLOTS[1]: 0
SLOTS[13]: 0
SLOTS[14]: 0
SLOTS[15]: 0
SLOTS[16]: 0
SLOTS[17]: 0
SLOTS[18]: 0
SLOTS[19]: 0
SLOTS[20]: 0
SLOTS[21]: 0
SLOTS[22]: 0
SLOTS[23]: 0
SLOTS[24]: 0
SLOTS[25]: 0
SLOTS[26]: 0
SLOTS[27]: 0
SLOTS[28]: 0
SLOTS[29]: 0
!!!!!!!!!!
Before reverse:
SLOTS[8]: (header)
SLOTS[9]: 80
SLOTS[10]: 10
SLOTS[11]: 60
SLOTS[12]: 40
```

4

```
190  SLOTS[7]: 70
191  After reverse:
192  SLOTS[8]: (header)
193  SLOTS[7]: 70
194  SLOTS[12]: 40
195  SLOTS[11]: 60
196  SLOTS[10]: 10
197  SLOTS[9]: 80
```

6 Grading Breakdown

TBD

7 Autograder Details

Once the autograder is set up on Gradescope, I will send a Canvas announcement and add additional details below.

UC**DAVIS**
COMPUTER SCIENCE