

Course : Programming Fundamental – ENSF 337

Lab # : Lab 8

Instructor : M. Moussavi

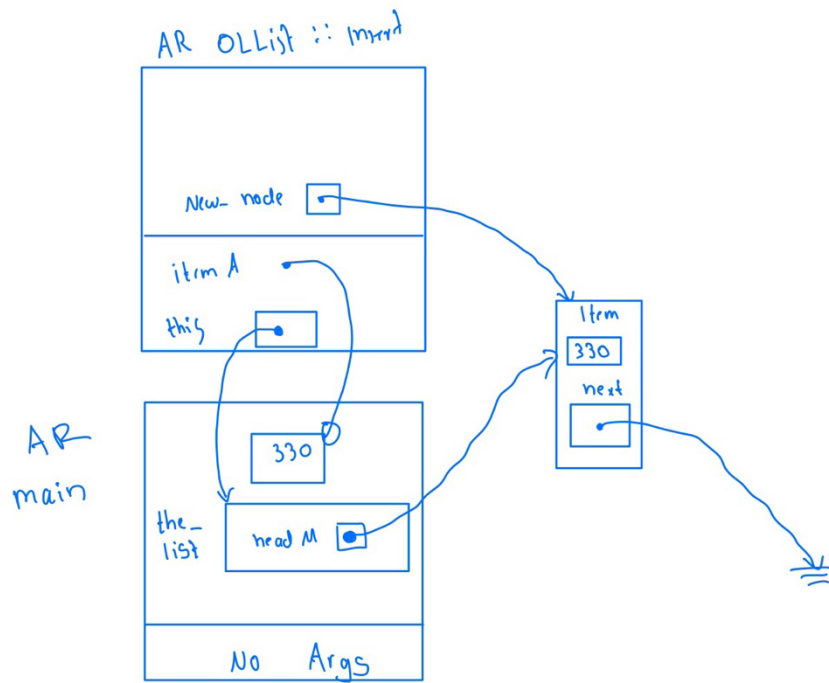
Student Name : Nimna Wijedasa

Lab Section : B02

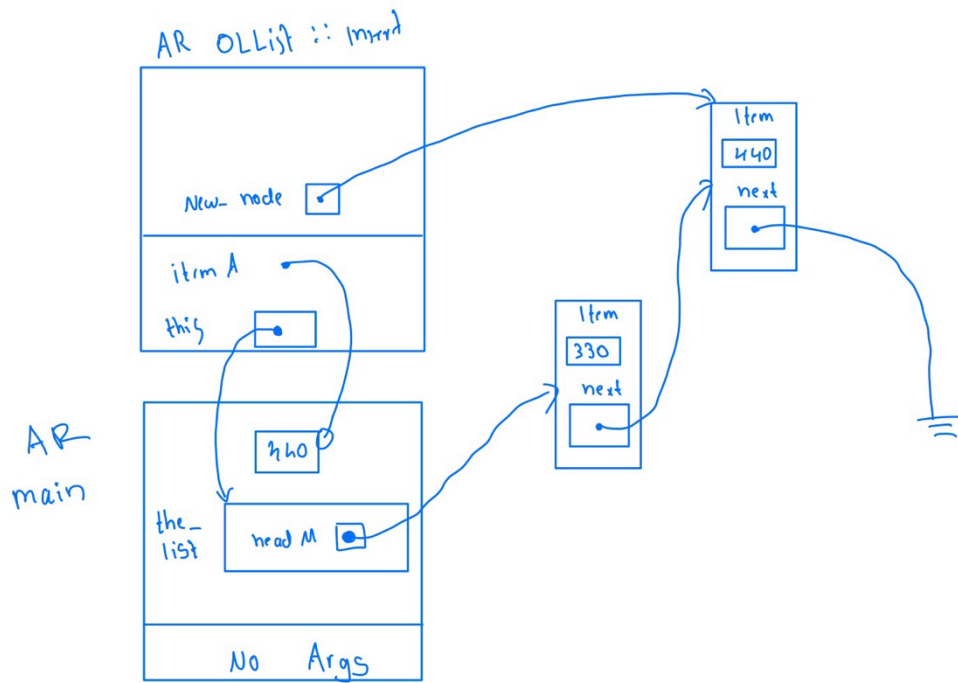
Date submitted : Nov 25, 2022

Exercise A

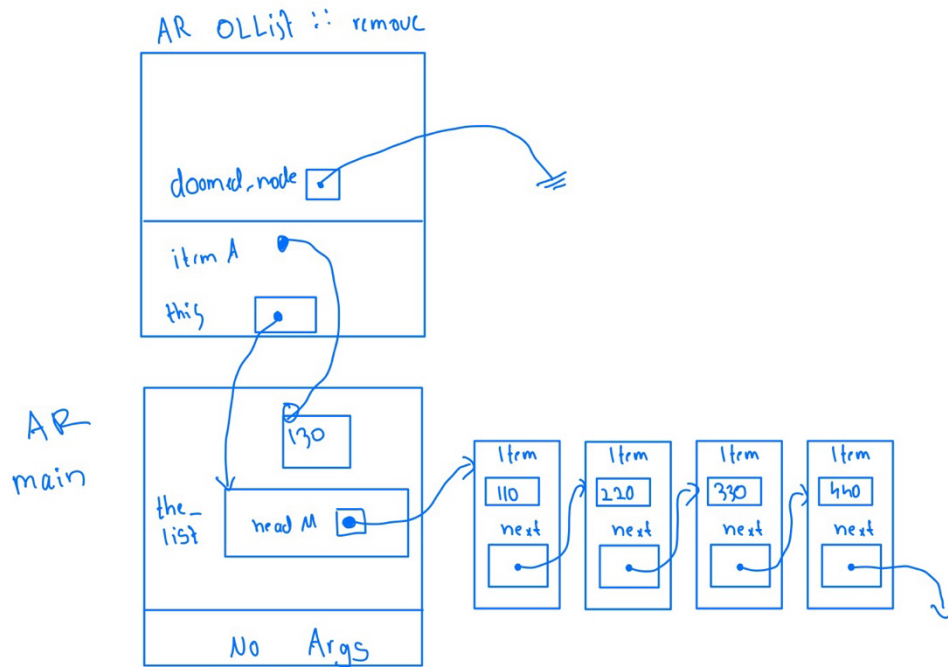
Point 1



Point 2



Point 3



Exercise B

```
lab8 x
/Users/nimnawijedasa/Desktop/fall/337/lab8/cmake-build-debug/lab8
List just after creation. expected to be [ ]
[ ]
the_list after some insertions. Expected to be: [ 99, 110, 120, 220, 330, 440, 550 ]
[ 99, 110, 120, 220, 330, 440, 550 ]
testing for copying lists ...
other_list as a copy of the_list: expected to be [ 99, 110, 120, 220, 330, 440, 550 ]
[ 99, 110, 120, 220, 330, 440, 550 ]
third_list as a copy of the_list: expected to be: [ 99, 110, 120, 220, 330, 440, 550 ]
[ 99, 110, 120, 220, 330, 440, 550 ]
testing for removing and chaining assignment operator...
the_list after some removals: expected to be: [ 99, 110, 120, 220, 440 ]
[ 99, 110, 120, 220, 440 ]
printing other_list one more time: expected to be: [ 99, 110, 120, 220, 330, 440, 550 ]
[ 99, 110, 120, 220, 330, 440, 550 ]
printing third_list one more time: expected to be: [ 99, 110, 120, 220, 330, 440, 550 ]
[ 99, 110, 120, 220, 330, 440, 550 ]
chaining assignment operator ...
the_list after chaining assignment operator: expected to be: [ 99, 110, 120, 220, 440 ]
[ 99, 110, 120, 220, 440 ]
other_list after chaining: expected to be: [ 99, 110, 120, 220, 440 ]
[ 99, 110, 120, 220, 440 ]
third_list after chaining: expected to be: [ 99, 110, 120, 220, 440 ]
[ 99, 110, 120, 220, 440 ]

Process finished with exit code 0
```

```
CMakeLists.txt x OLList.cpp x
1 // OLList.cpp
2 // ENSF 337 Fall 2021 Lab 8 Exercise A and B
3
4 #include <list>
5 using namespace std;
6 #include "OLList.h"
7
8
9 OLList::OLList()
10 : headM(0)
11 {
12 }
13
14 OLList::OLList(const OLList& source)
15 {
16     copy(source);
17 }
18
19 OLList& OLList::operator =(const OLList& rhs)
20 {
21     if (this != &rhs) {
22         destroy();
23         copy(source: rhs);
24     }
25     return *this;
26 }
27
28 OLList::~OLList()
29 {
30     destroy();
31 }
32
33 void OLList::print() const
34 {
35     cout << '[';
36     if (headM != 0) {
37         cout << ' ' << headM->item;
38         for (const Node *p = headM->next; p != 0; p = p->next)
39             cout << ", " << p->item;
40     }
41 }
```

OLList::copy

```
CMakeLists.txt x OLList.cpp x
40 }
41 cout << " ]\n";
42 }
43
44 void OLList::insert(const ListItem& itemA)
45 {
46     Node *new_node = new Node;
47     new_node->item = itemA;
48
49     if (headM == 0 || itemA <= headM->item ) {
50         new_node->next = headM;
51         headM = new_node;
52         // point one
53     }
54     else {
55         Node *before = headM; // will point to node in front of new node
56         Node *after = headM->next; // will be 0 or point to node after new node
57         while(after != nullptr && itemA > after->item) {
58             before = after;
59             after = after->next;
60         }
61         new_node->next = after;
62         before->next = new_node;
63         // point two
64     }
65 }
66
67 void OLList::remove(const ListItem& itemA)
68 {
69     // if list is empty, do nothing
70     if (headM == 0 || itemA < headM->item)
71         return;
72
73     Node *doomed_node = 0;
74
75     if (itemA == headM->item) {
76         doomed_node = headM;
77         headM = headM->next;
78     }
79 }
```

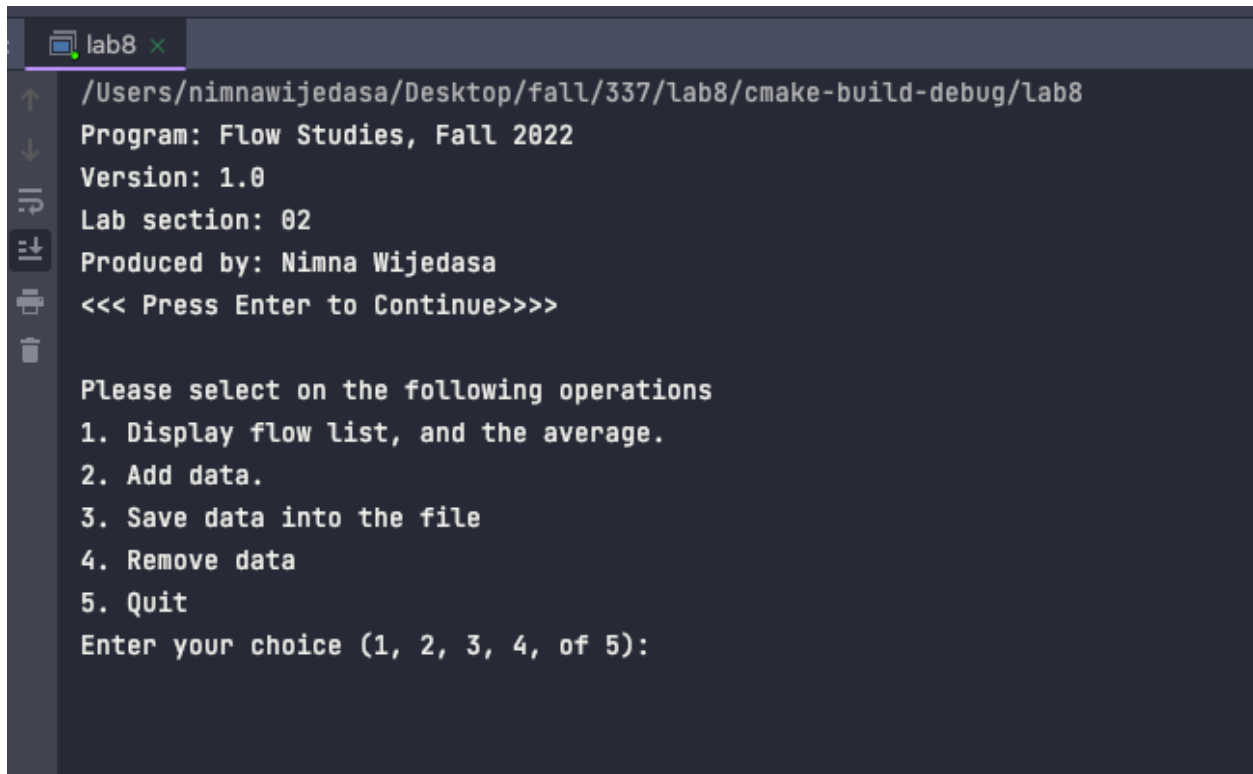
OLList::copy

```
CMakeLists.txt x OLList.cpp x
73     Node *doomed_node = 0;
74
75     if (itemA == headM->item) {
76         doomed_node = headM;
77         headM = headM->next;
78     }
79     else {
80         Node *before = headM;
81         Node *maybe_doomed = headM->next;
82         while(maybe_doomed != 0 && itemA > maybe_doomed->item) {
83             before = maybe_doomed;
84             maybe_doomed = maybe_doomed->next;
85         }
86         // point three
87         if (maybe_doomed != nullptr && maybe_doomed->item == itemA){
88             doomed_node = maybe_doomed;
89             before->next = maybe_doomed->next;
90         }
91     }
92     delete doomed_node;
93 }
94
95 void OLList::destroy()
96 {
97     Node *pointer = headM;
98     Node *ptr;
99     while (pointer != nullptr){
100         ptr = pointer;
101         pointer = pointer ->next;
102         delete ptr;
103     }
104     headM = nullptr;
105 }
106
107 void OLList::copy(const OLList& source)
108 {
109     if(source.headM == nullptr){
110         headM = nullptr;
111         return;
112     }
113     Node *new_node = new Node;
114     new_node->item = source.headM->item;
115     new_node->next = nullptr;
116     headM = new_node;
117     while(source.headM->next != nullptr){
118         new_node = new Node;
119         new_node->item = source.headM->next->item;
120         new_node->next = nullptr;
121         headM->next = new_node;
122         source.headM = source.headM->next;
123     }
124 }
```

OLList::copy

```
104     headM = nullptr;
105 }
106
107 void OLList::copy(const OLList& source)
108 {
109     if(source.headM == nullptr){
110         headM = nullptr;
111         return;
112     }
113     headM = new Node;
114     Node *temp_node = headM;
115     const Node *source_node = source.headM;
116     while(true){
117         temp_node->item = source_node->item;
118         source_node = source_node->next;
119         if( source_node == nullptr ){
120             break;
121         }
122         temp_node->next = new Node;
123         temp_node = temp_node->next;
124     }
125     temp_node->next = nullptr;
126 }
127
128
129
```


Exercise C



The screenshot shows a terminal window with a dark background. The title bar at the top indicates the window is named 'lab8'. The terminal output displays the following information:

```
/Users/nimnawijedasa/Desktop/fall/337/lab8/cmake-build-debug/lab8
Program: Flow Studies, Fall 2022
Version: 1.0
Lab section: 02
Produced by: Nimna Wijedasa
<<< Press Enter to Continue>>>>

Please select on the following operations
1. Display flow list, and the average.
2. Add data.
3. Save data into the file
4. Remove data
5. Quit
Enter your choice (1, 2, 3, 4, of 5):
```

On the left side of the terminal window, there is a vertical toolbar containing icons for navigation (up and down arrows), search (magnifying glass), and other standard terminal functions.

Please select on the following operations

1. Display flow list, and the average.
2. Add data.
3. Save data into the file.
4. Remove data..
5. Quit.

Enter your choice (1, 2, 3, 4, or 5):

1

Year	Flow
1970	100.34
1901	210.11
1947	310.99
1990	214.98
2002	211.44
1972	219.99
1900	220.11
1922	192.99
1945	145.66
1946	300.99
1971	209.99
1989	234.98
1999	110.99
2000	110.22
2001	231.44

The annual average of the flow is: 201.681 billions cubic metres

Please select on the following operations

1. Display flow list, and the average.
2. Add data.
3. Save data into the file.
4. Remove data..
5. Quit.

Enter your choice (1, 2, 3, 4, or 5):

2

Year: 1990

Flow: 234