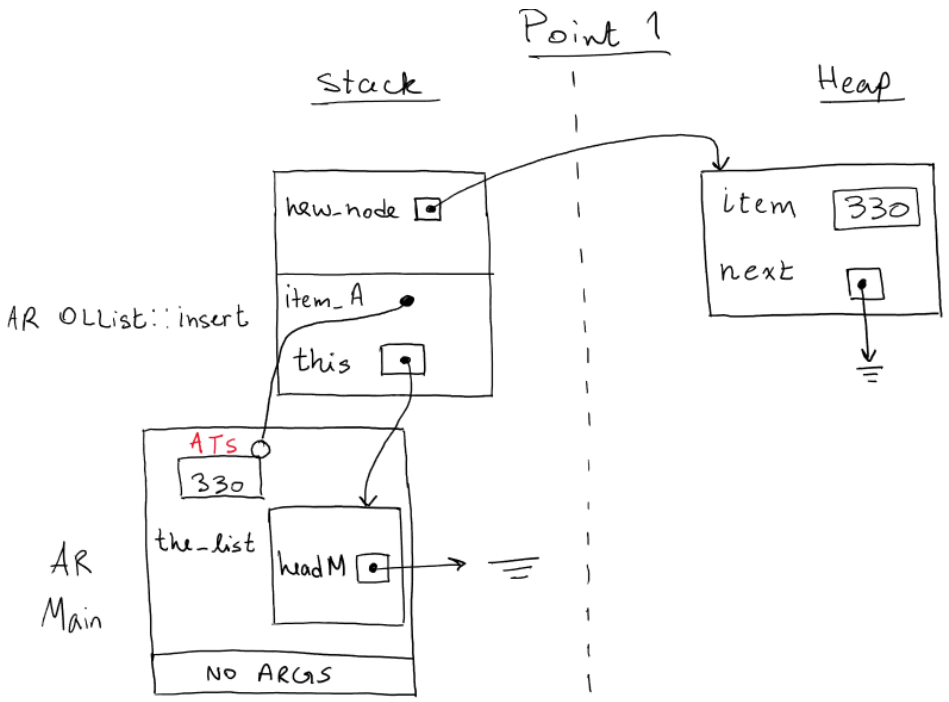
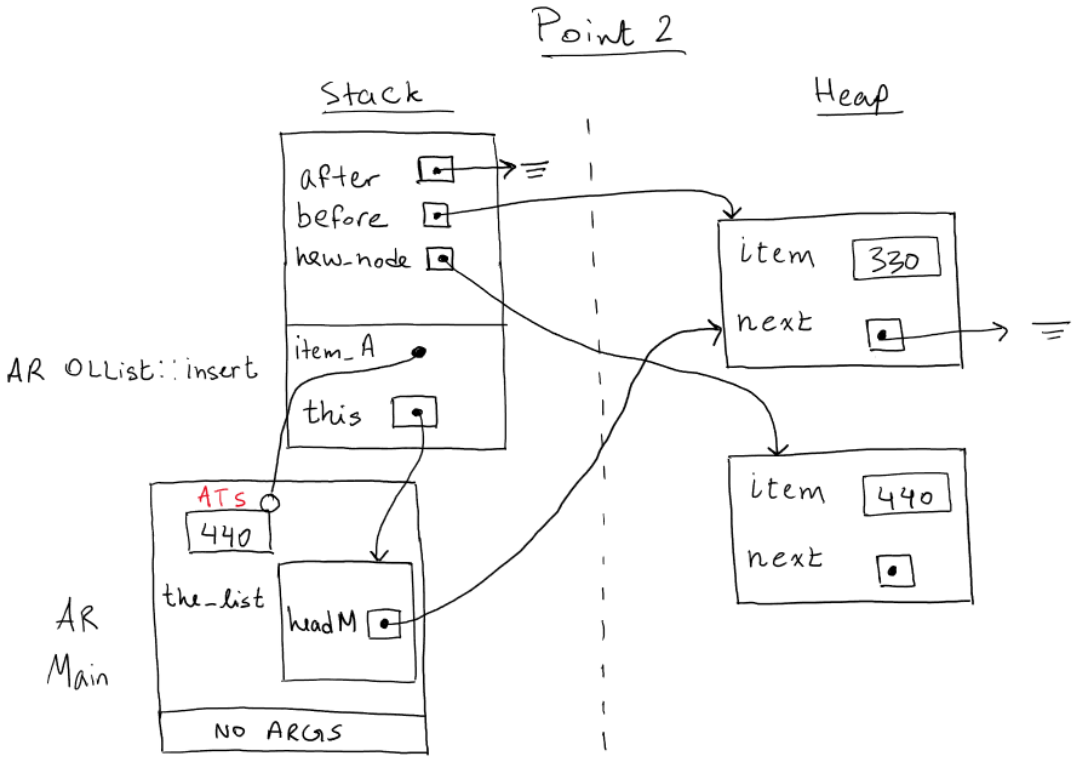
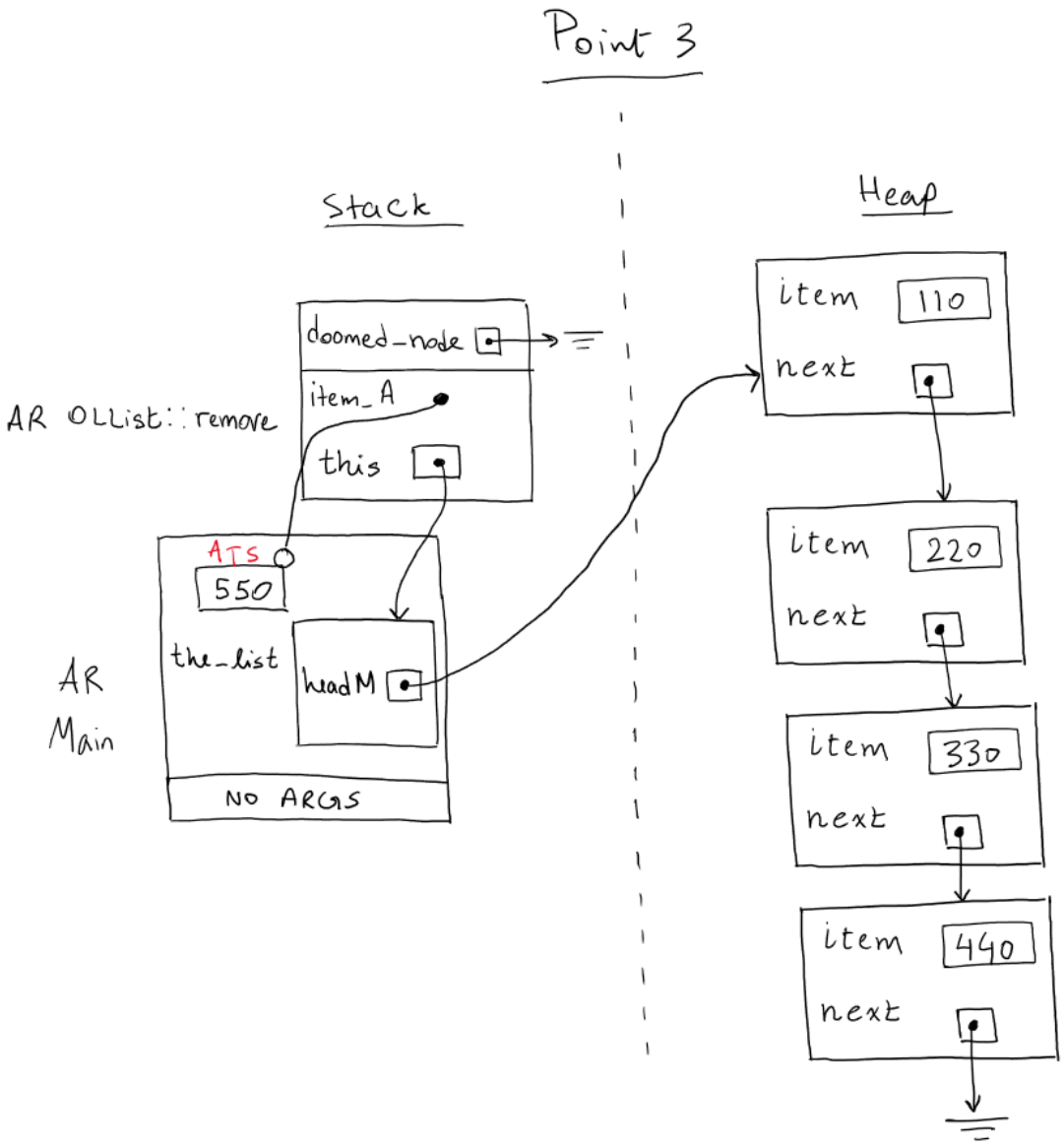
Course: Programming Fundamental – ENSF 337
Lab #: Lab 8
Instructor: M. Moussavi
Student Name: Aarushi Roy Choudhury
Lab Section: B01
Date submitted: Nov, 25 2021

Exercise A



ATS = Anonymous Temp Space




Point 3
Stack
Heap
AR OLList::remove
doomed_node
item_A
this
AR
Main
A_IS
550
the_list
headM
NO ARGS
item
110
next
item
220
next
item
330
next
item
440
next

Exercise B

```
// OLList.cpp
// ENSF 337 Fall 2021 Lab 8 Exercise A and B
// Completed by: Aarushi Roy Choudhury

#include <iostream>
#include <stdlib.h>

using namespace std;

#include "OLList.h"

OLList::OLList()
        : headM(0) {
}

OLList::OLList(const OLList &source) {
    copy(source);
}

OLList &OLList::operator=(const OLList &rhs) {
    if (this != &rhs) {
        destroy();
        copy(rhs);
    }
    return *this;
}

OLList::~OLList() {
    destroy();
}

void OLList::print() const {
    cout << '[';
    if (headM != 0) {
        cout << ' ' << headM->item;
        for (const Node *p = headM->next; p != 0; p = p->next)
            cout << ", " << p->item;
    }
    cout << " ]\n";
}

void OLList::insert(const ListItem &itemA) {
    Node *new_node = new Node;
```

```
    new_node->item = itemA;

    if (headM == 0 || itemA <= headM->item) {
        new_node->next = headM;
        headM = new_node;
// point one
    } else {
        Node *before = headM; // will point to node in front of new node
        Node *after = headM->next; // will be 0 or point to node after new node
        while (after != 0 && itemA > after->item) {
            before = after;
            after = after->next;
        }
        new_node->next = after;
        before->next = new_node;
// point two
    }
}

void OLList::remove(const ListItem &itemA) {
// if list is empty, do nothing
    if (headM == 0 || itemA < headM->item)
        return;

    Node *doomed_node = 0;


    if (itemA == headM->item) {

        doomed_node = headM;

        headM = headM->next;

        delete doomed_node;
    } else {

        Node *before = headM;
        Node *maybe_doomed = headM->next;
        while (maybe_doomed != 0 && itemA > maybe_doomed->item) {
            before = maybe_doomed;
            maybe_doomed = maybe_doomed->next;
        }

        if (maybe_doomed != nullptr) {
```

```
            before->next = maybe_doomed->next;

            delete maybe_doomed;
        }
// point three
    }
// the remaining part of this function is missing. As part of exercise B
// students are supposed to complete the rest of the definition of this function.
}

void OLList::destroy() {

// this function is not properly designed. As part of the exercise B
// students are supposed to remove the following lines and
// complete the definition of this helper function.


    Node* currentNode = headM;
    while (currentNode != nullptr) {

        Node* nextNode = currentNode->next;

        delete currentNode;

        currentNode = nextNode;
    }

    headM = nullptr;
}

void OLList::copy(const OLList &source) {
// this function is not properly designed. As part of the exercise B
// students are supposed to remove the following lines and
// complete the definition of this helper function.

// The only effect of the next line is to tell the compiler
// not to generate an "unused argument" warning. Don't leave it
// it in your solution.


    if (source.headM != nullptr) {

        Node *newNode = new Node;
        newNode->item = source.headM->item;
        newNode->next = nullptr;
```

```
        headM = newNode;


        Node *sourceTempNode = source.headM->next;
        Node *currentTempNode = headM;

        while (sourceTempNode != nullptr) {

            Node *newTempNode = new Node;

            newTempNode->item = sourceTempNode->item;

            newTempNode->next = nullptr;

            currentTempNode->next = newTempNode;

            currentTempNode = newTempNode;

            sourceTempNode = sourceTempNode->next;
        }
    } else {

        headM = nullptr;
    }
}
```

```
C:\Users\Aarus\Desktop\Lab8_A>g++ lab8ExB.cpp OLList.cpp

C:\Users\Aarus\Desktop\Lab8_A>a.exe
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_ist 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 ]

C:\Users\Aarus\Desktop\Lab8_A>
```

Exercise C

Please see the zip file for the code, outputs for each option are provided below. I also printed the data after insertions and deletions to show they were done successfully.

```
PS C:\Users\Aarus\Desktop\Lab8C> cd "c:\Users\Aarus\Desktop\Lab8C\" ; if (
Program: Flow Studies - Fall 2021
Version: 1.0
Lab section: B01
Produced by: Aarushi Roy Choudhury
Please select one of 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):1
Year  Flow
1900   220.11
1901   210.11
1922   192.99
1945   145.66
1946   300.99
1947   310.99
1970   100.34
1971   209.99
1972   219.99
1989   234.98
1990   214.98
1999   110.99
2000   110.22
2001   231.44
2002   211.44
The annual average of the flow is: 201.681 billions of cubic meter.

<<< Press Enter to Continue >>>
```

```
Please select one of 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):2
Please enter a year: 2021
Please enter the flow: 235.67
New record inserted successfully.

<<< Press Enter to Continue >>>
Please select one of 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):1
Year  Flow
1900   220.11
1901   210.11
1922   192.99
1945   145.66
1946   300.99
1947   310.99
1970   100.34
1971   209.99
1972   219.99
1989   234.98
1990   214.98
1999   110.99
2000   110.22
2001   231.44
2002   211.44
2021   235.67
The annual average of the flow is: 203.806 billions of cubic meter.

<<< Press Enter to Continue >>>
```

```
Please select one of 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):4
Please enter the year that you want to remove: 2002
Record was successfully removed.

<<< Press Enter to Continue >>>
Please select one of 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):1
Year  Flow
1900   220.11
1901   210.11
1922   192.99
1945   145.66
1946   300.99
1947   310.99
1970   100.34
1971   209.99
1972   219.99
1989   234.98
1990   214.98
1999   110.99
2000   110.22
2001   231.44
2021   235.67
The annual average of the flow is: 203.297 billions of cubic meter.

<<< Press Enter to Continue >>>
Please select one of 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):5

Program Terminated.

PS C:\Users\Aarus\Desktop\Lab8C>
```