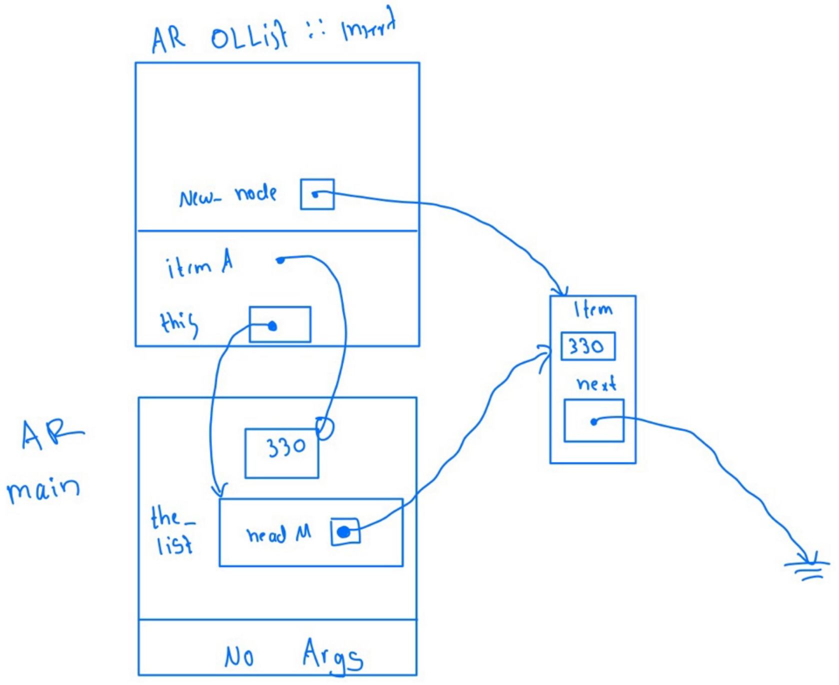
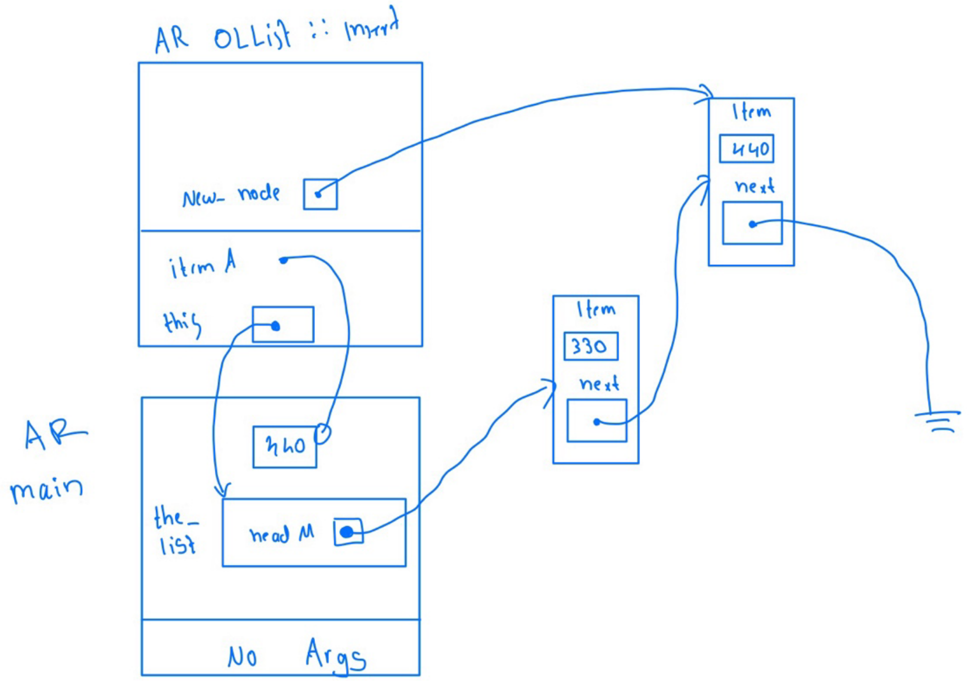
Course	: Programming Fundamental – ENSF 337
Lab #	: Lab 8
Instructor	: M. Moussavi
Student Name	: Nimna Wijedasa
Lab Section	: B02
Date submitted	: Nov 25, 2022

Exercise A

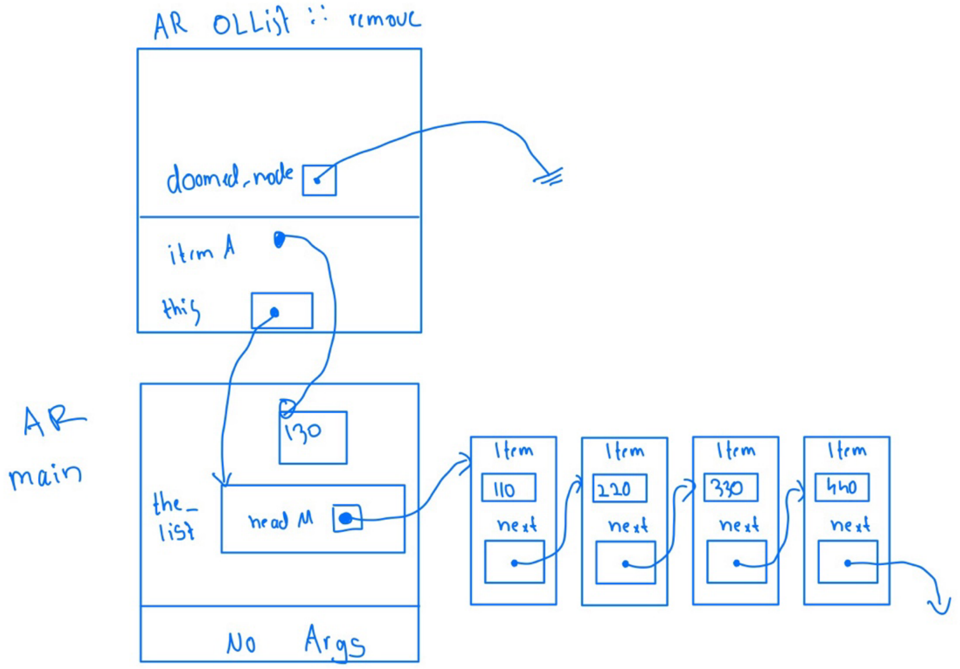
Point 1



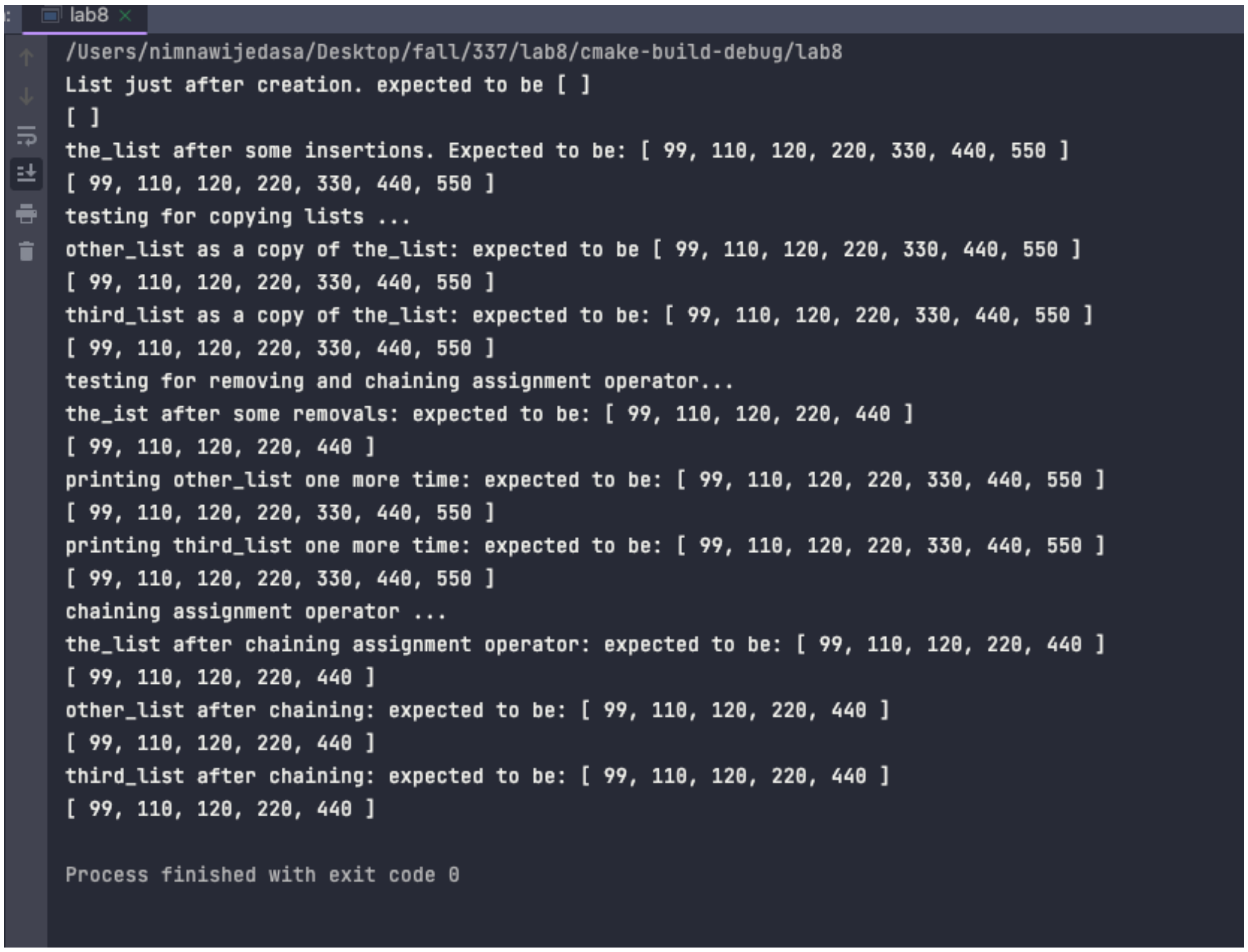
Point 2



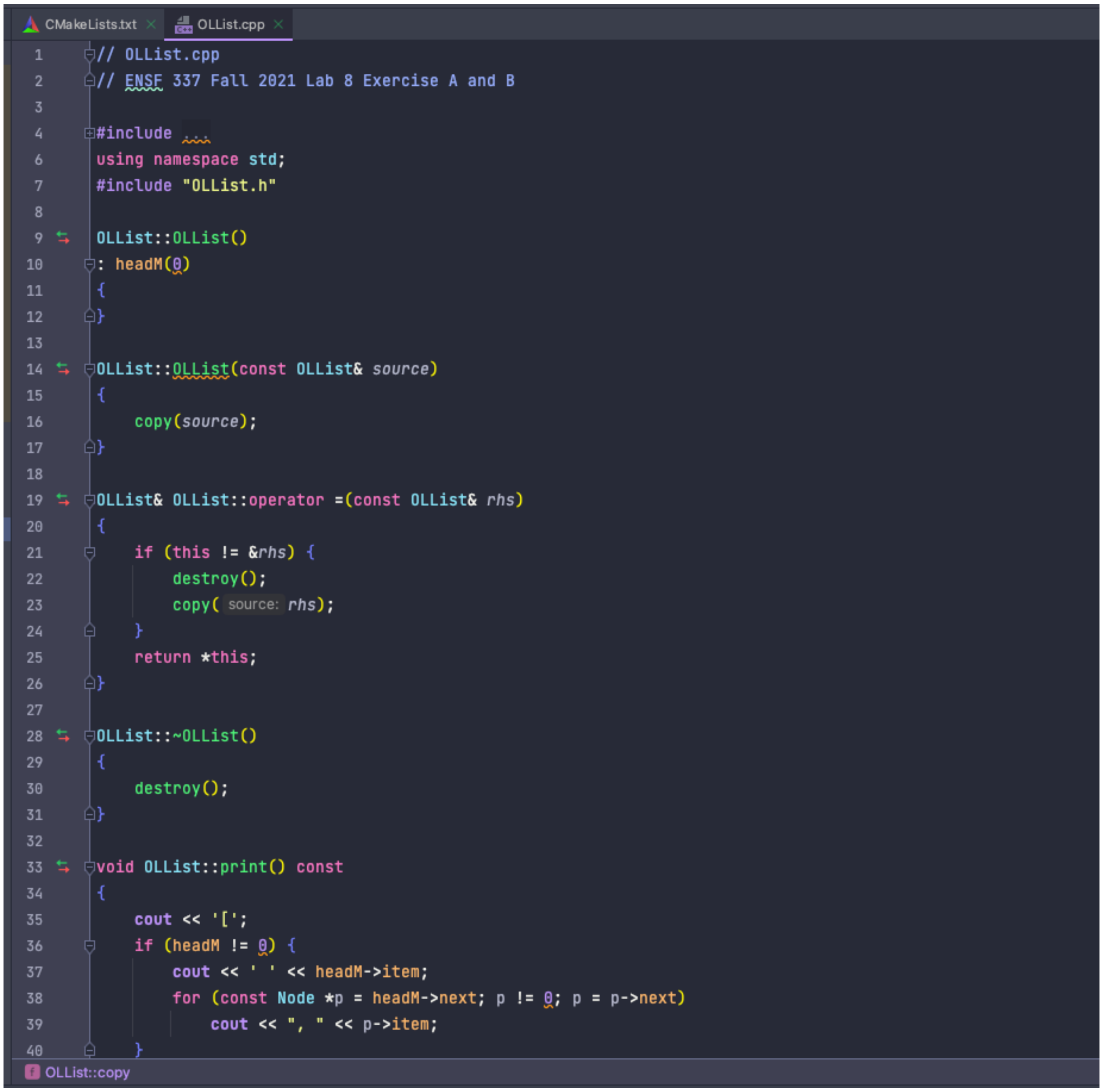
Point 3



Exercise B

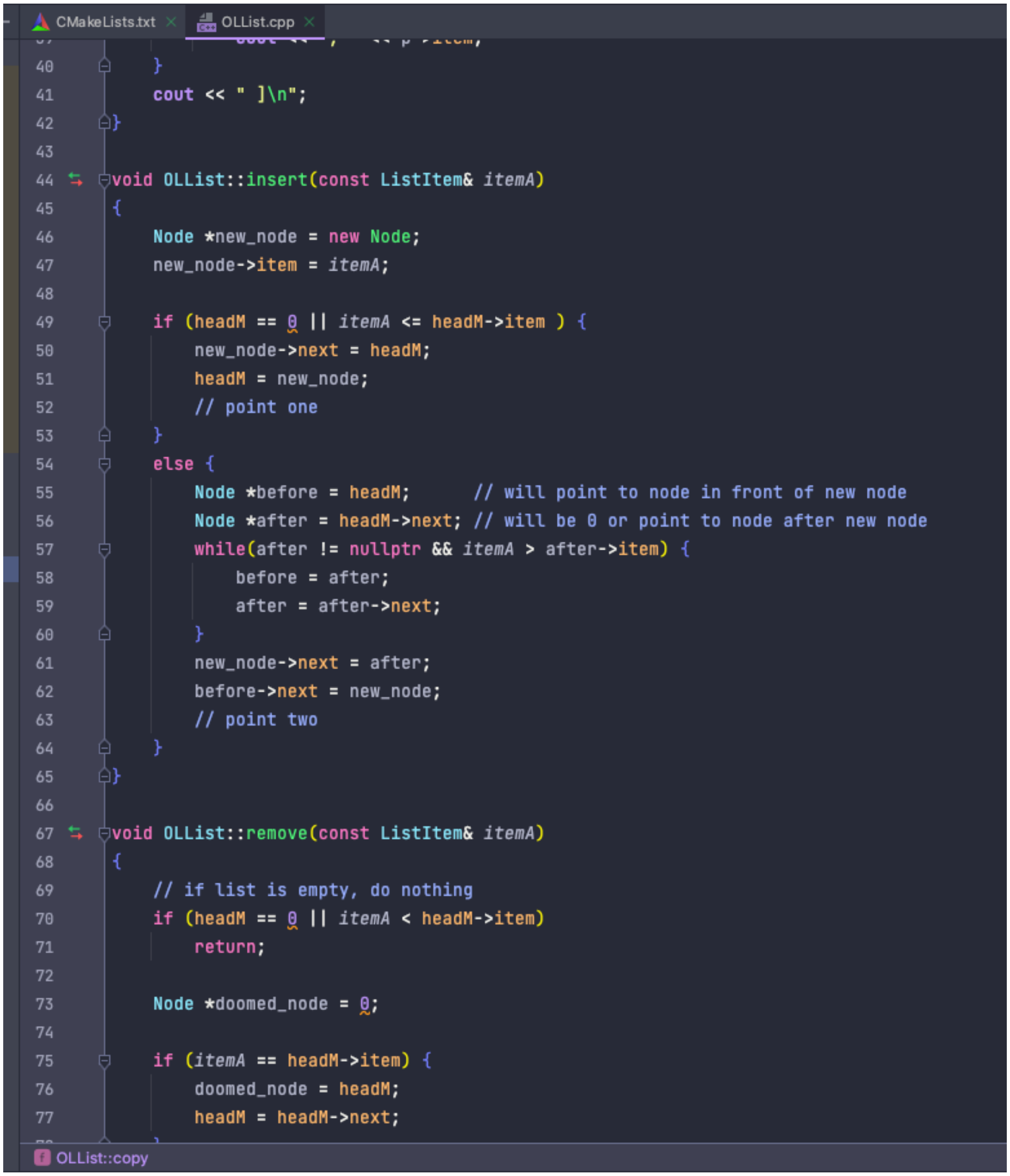

lab8

```
/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_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 ]

Process finished with exit code 0
```

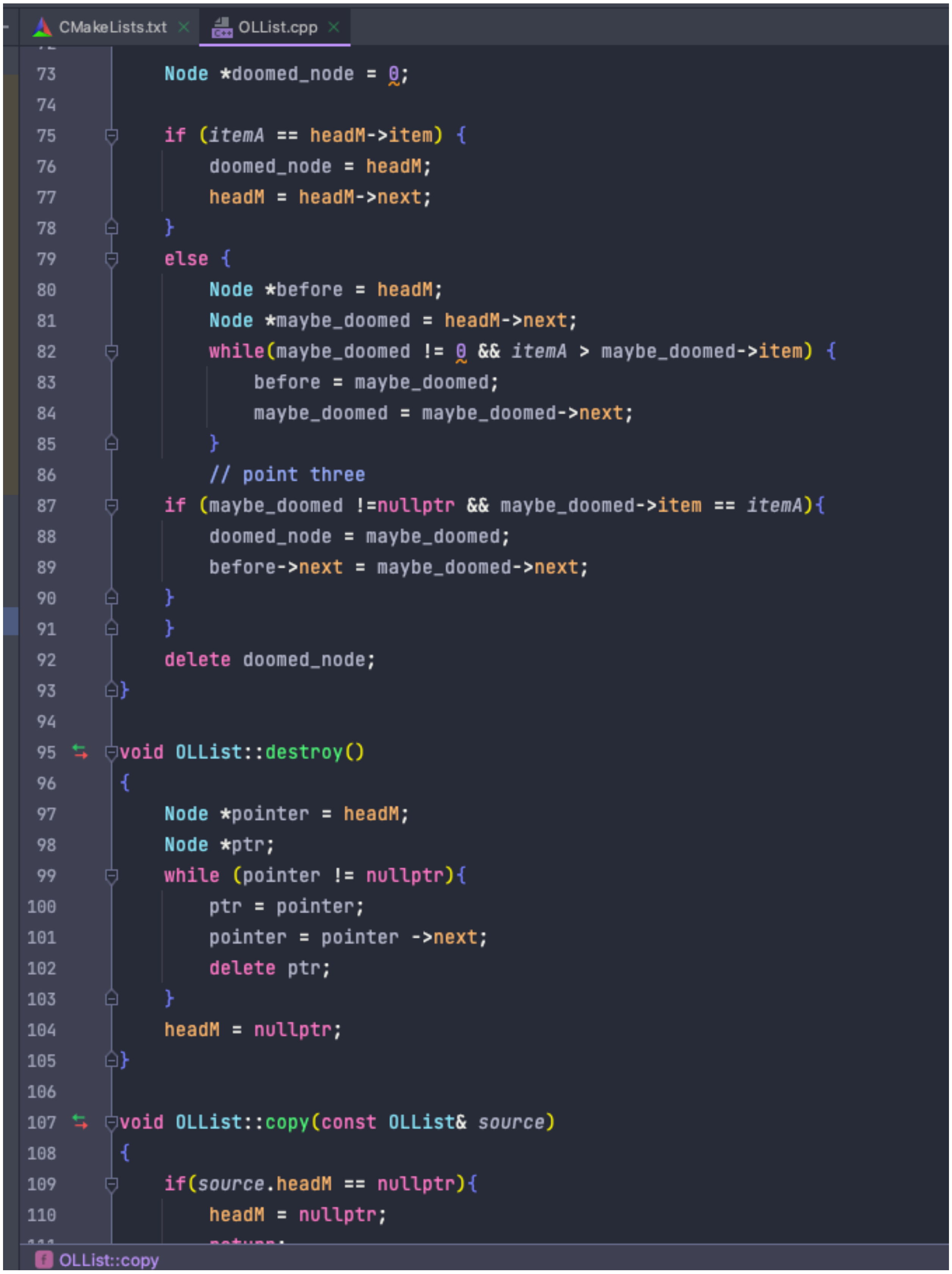

CMakeLists.txt OLList.cpp

```cpp
// OLList.cpp
// ENSF 337 Fall 2021 Lab 8 Exercise A and B

#include ...
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( source: 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;
    }
```

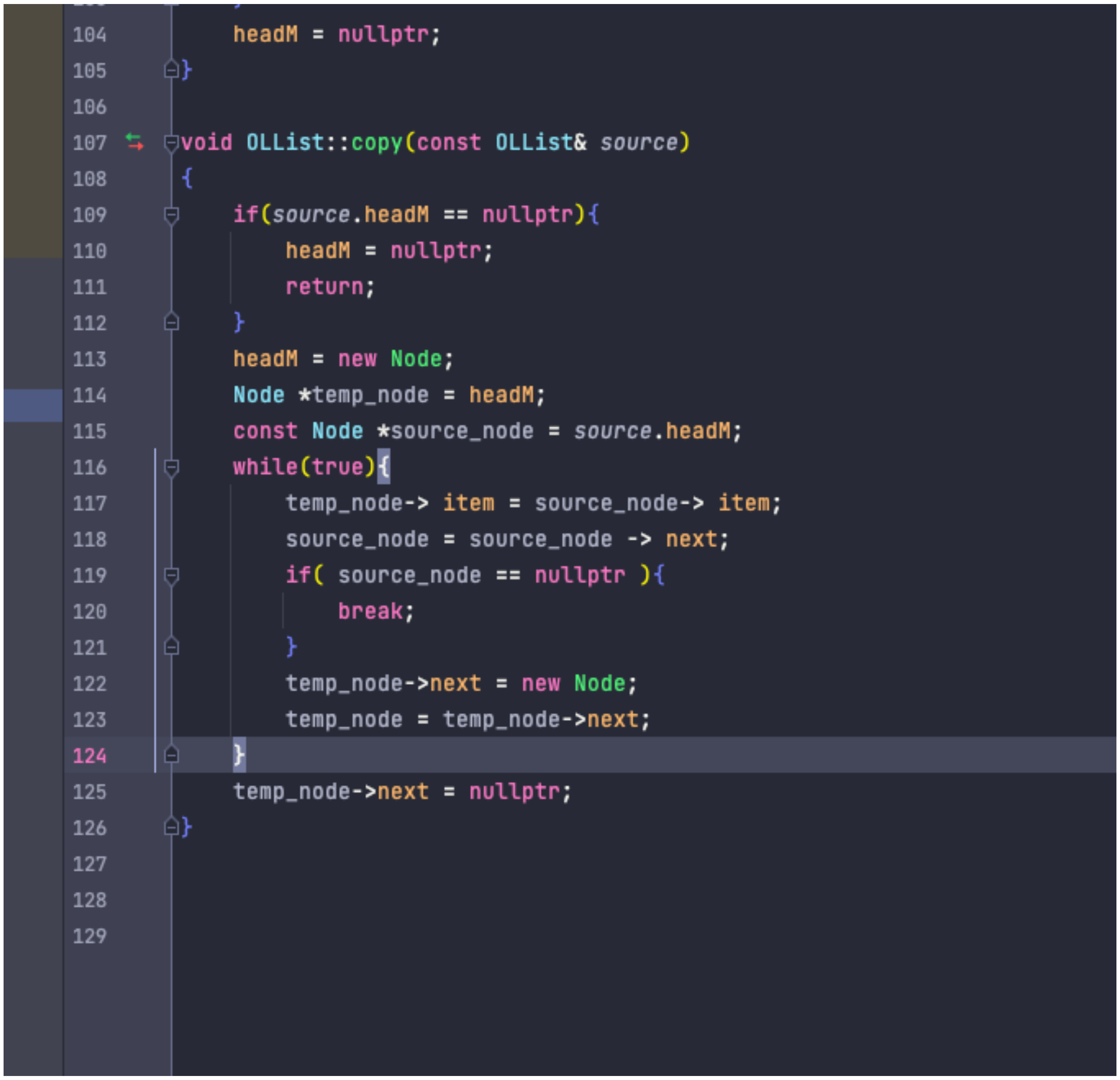
OLList::copy



```
    }
    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 != nullptr && 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;
```

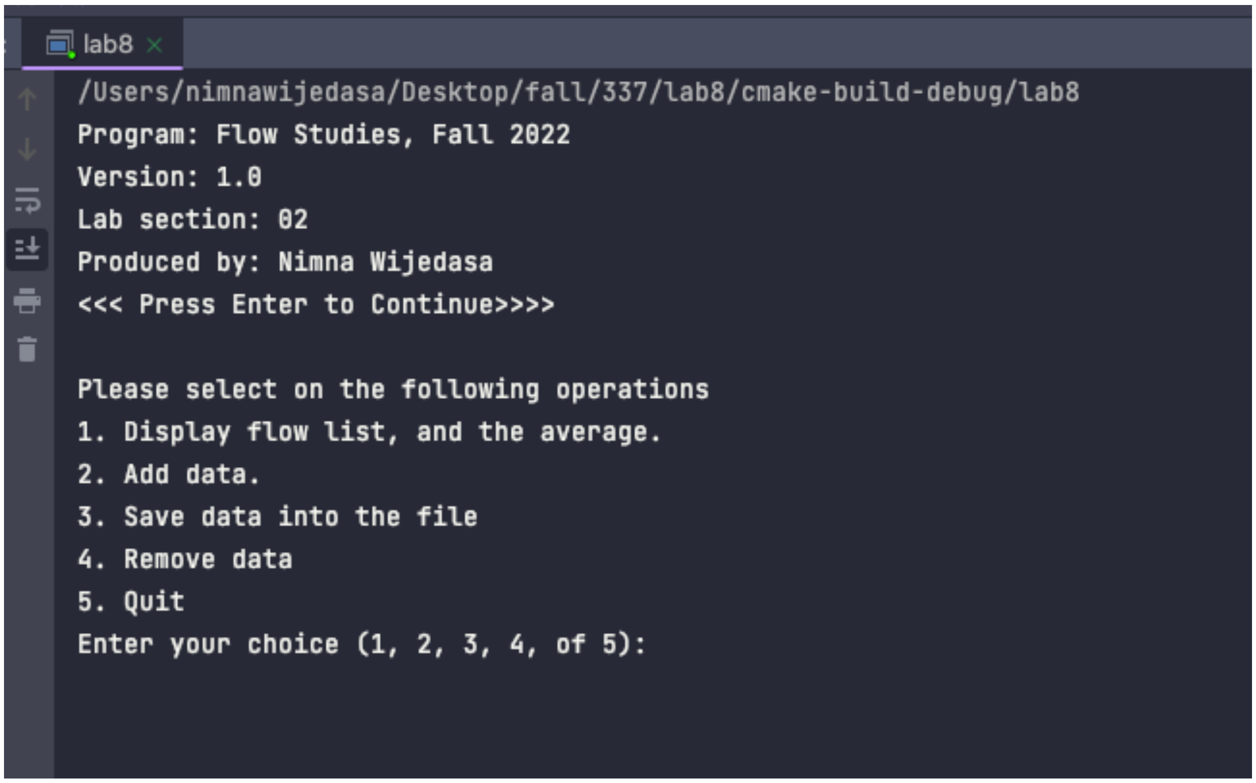


```cpp
    Node *doomed_node = 0;

    if (itemA == headM->item) {
        doomed_node = headM;
        headM = headM->next;
    }
    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;
        }
        // point three
    if (maybe_doomed !=nullptr && maybe_doomed->item == itemA){
        doomed_node = maybe_doomed;
        before->next = maybe_doomed->next;
    }
    }
    delete doomed_node;
}

void OLList::destroy()
{
    Node *pointer = headM;
    Node *ptr;
    while (pointer != nullptr){
        ptr = pointer;
        pointer = pointer ->next;
        delete ptr;
    }
    headM = nullptr;
}

void OLList::copy(const OLList& source)
{
    if(source.headM == nullptr){
        headM = nullptr;
```

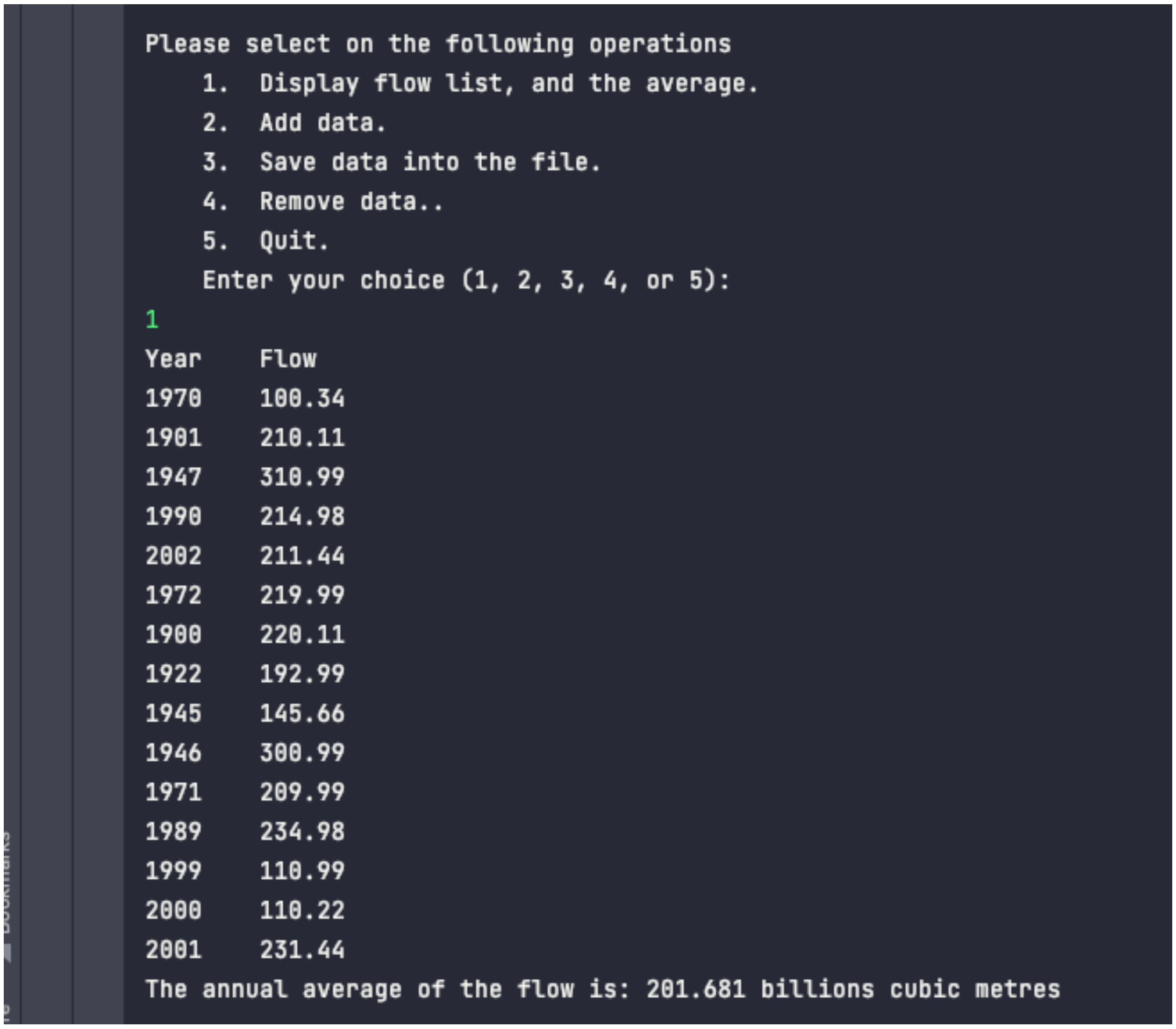


```
    headM = nullptr;
}

void OLList::copy(const OLList& source)
{
    if(source.headM == nullptr){
        headM = nullptr;
        return;
    }
    headM = new Node;
    Node *temp_node = headM;
    const Node *source_node = source.headM;
    while(true){
        temp_node-> item = source_node-> item;
        source_node = source_node -> next;
        if( source_node == nullptr ){
            break;
        }
        temp_node->next = new Node;
        temp_node = temp_node->next;
    }
    temp_node->next = nullptr;
}
```

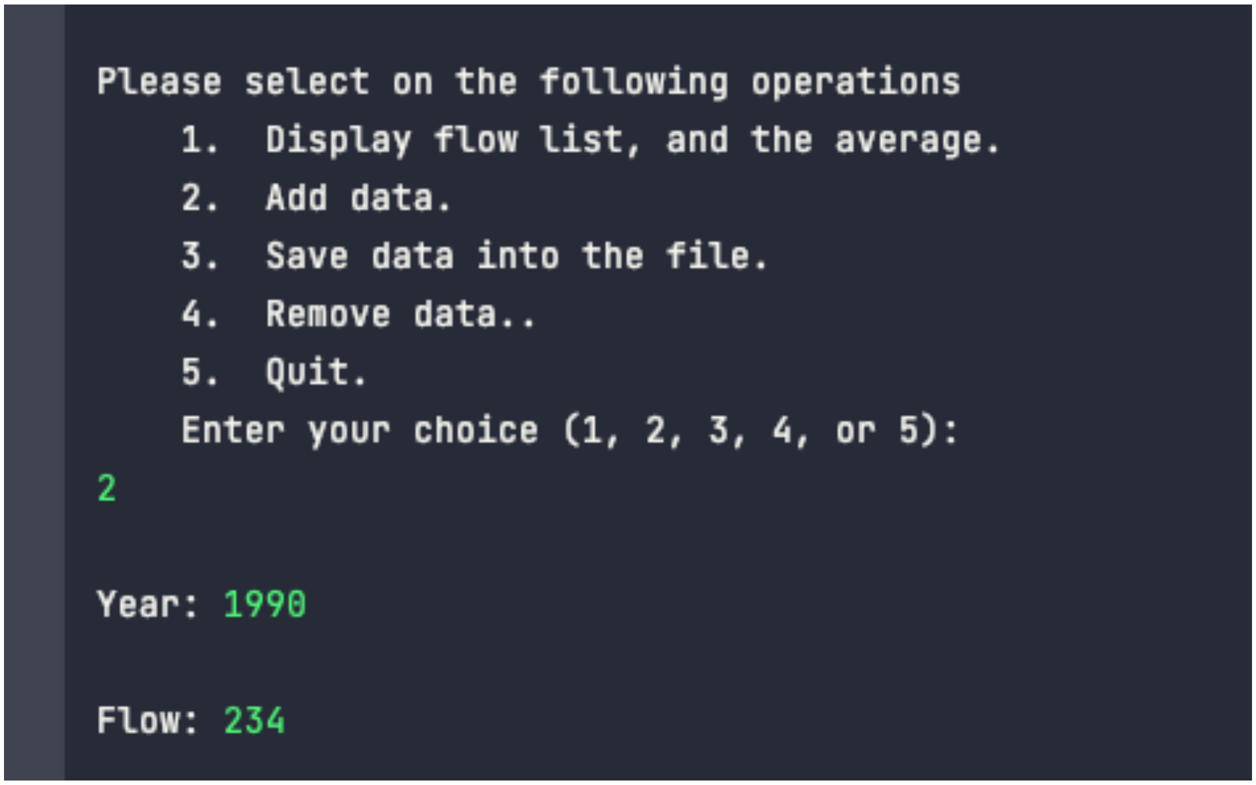
Exercise C



```
/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):
```



```
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
```