Programming Fundamentals – ENSF 337

Lab 8

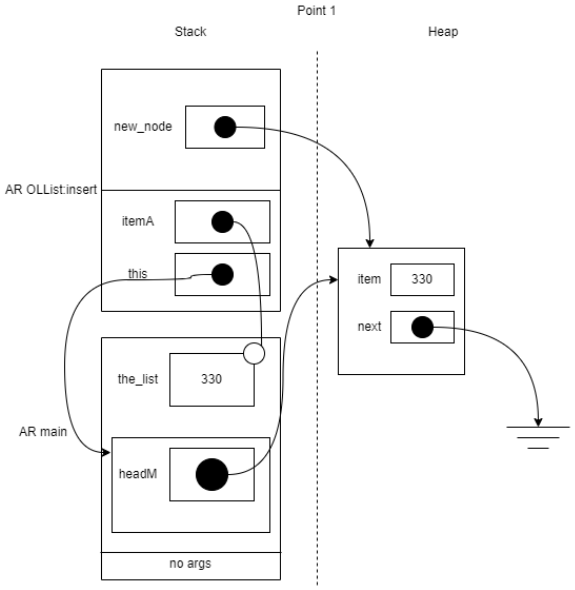
M. Moussavi

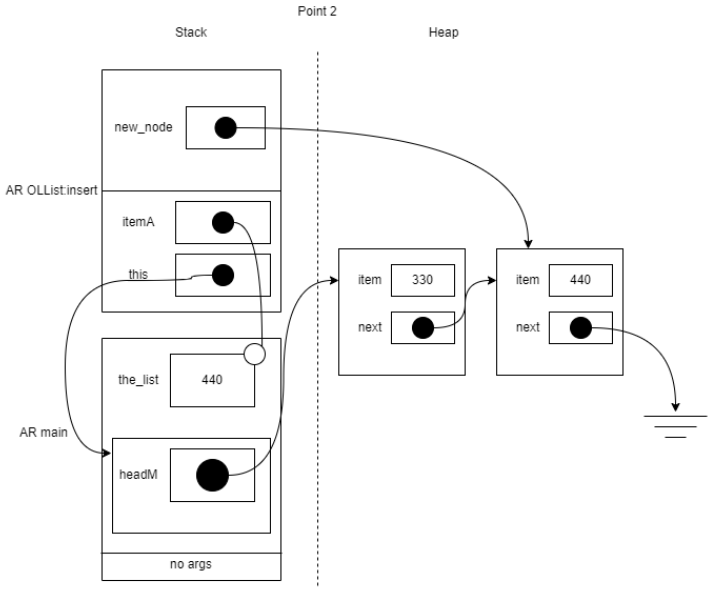
Jaxon Braun

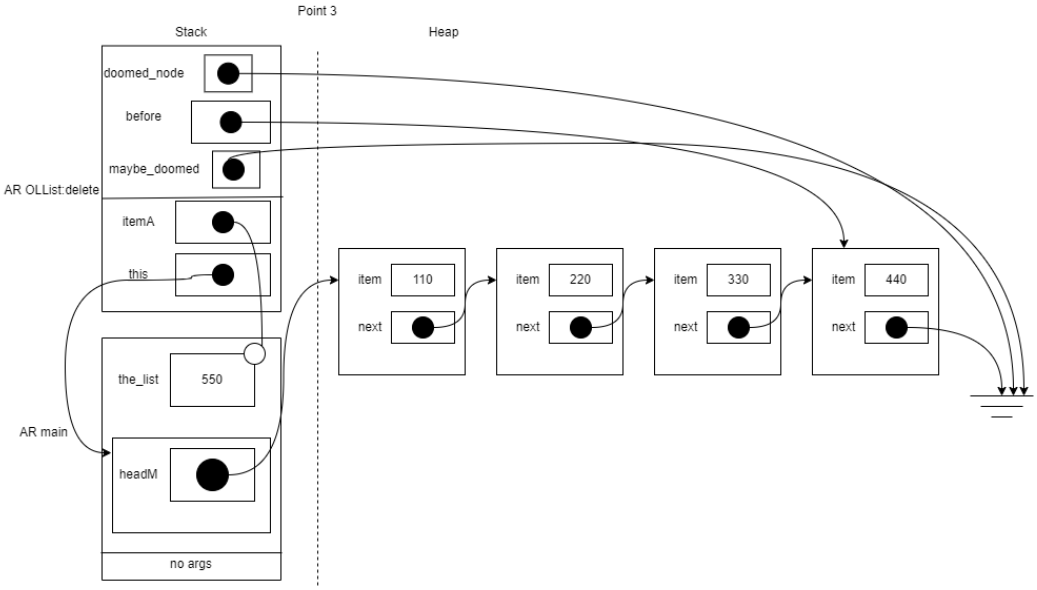
B01

Submitted on November 24, 2021

Exercise A:


Point 1
Stack
Heap
new_node
AR OLList:insert
itemA
this
item
330
next
the_list
330
AR main
headM
no args


Point 2
Stack
Heap
new_node
AR OLList:insert
itemA
this
item
330
next
item
440
next
the_list
440
AR main
headM
no args


Point 3
Stack
Heap
doomed_node
before
maybe_doomed
AR OLList:delete
itemA
this
item
110
next
item
220
next
item
330
next
item
440
next
the_list
550
AR main
headM
no args

Exercise B:

```
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 ]
```

Exercise C:

Source Code:

```
//list.h
//ENSF 337Lab 8 Exercise C
//Created by: Jaxon Braun

#ifndef LIST_H
#define LIST_H

struct ListItem   //struct to hold data at each node in linked list
{
   int year;
   double flow;
};

struct Node          //node used to store every element of struct ListItem in linked list
{
   ListItem item;
   Node *next;
};

class FlowList{
public:
   int record_count;
   Node *ptr;
   FlowList(); // PROMISES: Creates empty list.

   void insert(const struct ListItem& itemA);
   // PROMISES:
   //    A node with a copy of itemA is added in
   //    a way that preserves the nondecreasing
   //    order of items in nodes.

   void remove(const struct ListItem& itemA);
```

```
  //  PROMISES:
  //    If no node has an item matching itemA,
  //    list is unchanged.
  //    Otherwise exactly one node with
  //    its item == itemA is removed.

  void print() const;
  // PROMISES:
  //  Prints data in table format without headers

  int sum();
  // PROMISES:
  //  Calculates and returns the sum of the flow data

  int searchForYear(ListItem& itemA);
  // PROMISES:
  //  Returns 1 if year in argument exits in FlowList, 0 otherwise

private:
  Node *headM;
};

#endif
```

```
//list.cpp
//ENSF 337 Lab 8 Exercise C
//Creathed by: Jaxon Braun

#include <stdlib.h>
#include <iostream>
using namespace std;
#include "list.h"

FlowList::FlowList() //initiales empty linked list
: headM(0), ptr(0), record_count(0)
{
}

void FlowList::insert(const ListItem& itemA)
{
   Node *new_node = new Node;    //Creating the new node from given data itemA
   new_node->item.year = itemA.year;
   new_node->item.flow = itemA.flow;

   if (headM == 0 || itemA.year <= headM->item.year ) { //checks if new node should go in first position
      new_node->next = headM;
      headM = new_node;
      ptr = headM;
   }
   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.year > after->item.year) {
         before = after;
         after = after->next;
      }
      new_node->next = after;
      before->next = new_node;  //moves pointers to correct positions to finalize the insertion
   }
```

```
   record_count += 1;
}

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

   Node *doomed_node = 0;

   if (itemA.year == headM->item.year) { //checks if node to be deleted is the first one
      doomed_node = headM;
      headM = headM->next;
      ptr = headM;
   }
   else {                        //searches for the node to be removed from the list
      Node *before = headM;
      Node *maybe_doomed = headM->next;
      while(maybe_doomed != 0 && itemA.year > maybe_doomed->item.year) {
         before = maybe_doomed;
         maybe_doomed = maybe_doomed->next;
      }
      doomed_node = maybe_doomed;
      before->next = maybe_doomed->next;
      delete doomed_node;
   }
   record_count -= 1;
}

void FlowList::print() const
{
   if (headM != 0) {
      cout << headM->item.year << "\t" << headM->item.flow << endl;
      for (const Node *p = headM->next; p != 0; p = p->next)
```

```
        cout << p->item.year << "\t" << p->item.flow << endl;
    }
}

int FlowList::sum()
{
    int sum = 0;
    if (headM != 0){
        sum += headM->item.flow;
        for (const Node *p = headM->next; p != 0; p = p->next)
            sum += p->item.flow;
    }
    return sum;
}

int FlowList::searchForYear(ListItem& itemA)
{
    if (headM->item.year == itemA.year)
            return 1;
    else{
        for (const Node *p = headM->next; p != 0; p = p->next){
        if (headM->item.year == itemA.year)
            return 1;
        if (p->item.year == itemA.year)
            return 1;
        }
    }
    return 0;
}
```

```
//hydro.h
//ENSF 337 Lab 8 Exercise C
//Created by: Jaxon Braun

#ifndef HYDRO_H
#define HYDRO_H
#include "list.h"

void displayHeader();
//PROMISES: Displays introductory header

void readData(const char* infile, FlowList &data);
//PROMISES: Reads data from flow.txt text file
//        Places data into linked list in ascending order by year
//        by using FlowList member function insert
//        Updates the record_count, which is part of the FlowList object
//        I felt having the record_count be a part of the FlowList Object made more sense

int menu();
//Promises: To display the main menu and return the user's choice

void display(FlowList &data);
//Promises: Displays years and flows in the linked list
//        shows the average of the flow in the list

void addData(FlowList &data);
//Promises: prompts user for new data entry
//        inserts data into list
//        updates the number of records

void removeData(FlowList &data);
//Promises: prompts user for the year they would like removed from list
//        only removes a single record
//        updates the number of records
```

```
double average(FlowList &data);

//Promises: calculates and returns the average flow for records in list


void saveData(const char* outfile, FlowList &data);

//Promises: opens flow.txt file

//        writes the data from the list into the text file


void pressEnter();

//Promises: Displays the <<<<Press Enter to Continue>>>>, and waits for user input

//        returns after user input <Return Key> is detected


#endif
```

```
//hydro.cpp
//ENSF 337 Lab 8 Exercise C
//Created by: Jaxon Braun

#include <stdlib.h>
#include <iostream>
#include <fstream>
using namespace std;
#include "hydro.h"
#include "list.h"
#define UNDERLINE "\033[4m"
#define CLOSEUNDERLINE "\033[0m"

int main(){
   FlowList data;
   const char* infile = "flow.txt"; //will need to alter file name if flow.txt is in a diectory different from directory program is compilled in at terminal
   displayHeader();
   readData(infile, data);

   int quit = 0;

   while (1)
   {
      switch(menu()){
         case 1:
            display(data);
            pressEnter();
            break;
         case 2:
            addData(data);
            pressEnter();
            break;
         case 3:
            saveData(infile, data);
```

```
            pressEnter();
            break;
        case 4:
            removeData(data);
            pressEnter();
            break;
        case 5:
            cout << "\nProgram Terminated!\n\n";
            quit = 1;
            break;
        default:
            cout << "\nNot a Valid Input.\n";
            pressEnter();

    }
    if(quit == 1) break;
  }

  return 0;
}

void pressEnter(){
  cout << "\n<<<< Press Enter to Continue >>>>\n";
  cin.get();
  cin.ignore();   //added so it would actually pause for input
            //after first pressEnter call in main, but
            //now 2 enter presses are required on first use of pressEnter()
}

void displayHeader(){
  cout << "Program: Flow Studies - Fall 2021" << endl;
  cout << "Version: 1.0" << endl;
  cout << "Lab Section: B01" << endl;
  cout << "Produced by: Jaxon Braun" << endl;
  pressEnter();
```

```
}

int menu(){
   int user_input = 0;
   cout << "\nPlease Select An Operation:" << endl;
   cout << "1.\tDisplay Flow List, and the Average" << endl;
   cout << "2.\tAdd Data" << endl;
   cout << "3.\tSave Data Into the File" << endl;
   cout << "4.\tRemove Data" << endl;
   cout << "5.\tQuit" << endl;
   cout << "Enter Your Choice (1, 2, 3, 4, or 5):" << endl;
   cin >> user_input;
   return user_input;

}

void readData(const char* infile, FlowList &data){
   int year_tmp = 0;
   int flow_tmp = 0;
   ListItem ListItem_tmp;
   ifstream inData;
   inData.open(infile);
   if (inData.fail()) {
      cout << "Error: cannot open the file" << infile << endl;
      exit(1);
   }
   while (! inData.eof()){
      inData >> ListItem_tmp.year >> ListItem_tmp.flow;
      data.insert(ListItem_tmp);
   }
   inData.close();
}

double average(FlowList &data){
   double sum = 0;
```

```
    if (data.ptr != 0){
        for (const Node *p = data.ptr; p != 0; p = p->next)
            sum += p->item.flow;
    }
    double ave = sum / data.record_count;
    return ave;
}

void display(FlowList &data){
    cout << UNDERLINE << "Year\t" << CLOSEUNDERLINE << UNDERLINE << "Flow (in billions of cubic meters)" << CLOSEUNDERLINE << endl;
    data.print();
    cout <<  "\nNumber of Data Points: " << data.record_count;
    cout << "\nThe annual average of the flow is: " << average(data) << " billions of cubic meters" << endl;
}

void addData(FlowList &data){
    ListItem new_data_point;
    cout << "Please Enter a Year: ";
    cin >> new_data_point.year;
    cout << "Please Enter the flow: ";
    cin >> new_data_point.flow;
    if (data.searchForYear(new_data_point)){
        cout << "Error: Duplicate Data" << endl;
    }
    else{
        data.insert(new_data_point);
        cout << "New Record Inserted Successfully" << endl;
    }
}

void removeData(FlowList &data){
    ListItem doomed_data_point;
    cout << "Please Enter a Year to Remove it's Record: ";
    cin >> doomed_data_point.year;
    if (data.searchForYear(doomed_data_point)){
```

```
        data.remove(doomed_data_point);
        cout << "Record was successfully removed" << endl;
    }
    else
        cout << "Error: No such data with that year on record" << endl;
}

void saveData(const char* outfile, FlowList &data){
    ofstream outData;
    outData.open(outfile);
    if (! outData){
        cout << "Error: cannot open the file " << outfile << endl;
        return;
    }
    for (const Node *p = data.ptr; p != 0; p = p->next)
        outData << p->item.year << "\t" << p->item.flow << endl;
    outData.close();
    cout << "Data is saved into the file " << outfile << endl;
}
```

Terminal Outputs:

Output after launching executable and pressing enter

```
Program: Flow Studies - Fall 2021
Version: 1.0
Lab Section: B01
Produced by: Jaxon Braun

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



Please Select An Operation:
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):
```

Output after entering 1

```
Enter Your Choice (1, 2, 3, 4, or 5):
1
Year    Flow (in billions of cubic meters)
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

Number of Data Points: 15
The annual average of the flow is: 201.681 billions of cubic meters

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

Output after pressing enter

```
Please Select An Operation:
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):
```

Output after pressing 2 and adding new data points, two that are new and one that already exists

```
Please Select An Operation:
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
Please Enter a Year: 1969
Please Enter the flow: 321.33
New Record Inserted Successfully

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

```
Please Select An Operation:
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
Please Enter a Year: 2002
Please Enter the flow: 100.24
Error: Duplicate Data

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

Output after pressing 4 and removing data points, one that exists and one that does not

```
Please Select An Operation:
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):
4
Please Enter a Year to Remove it's Record: 1900
Record was successfully removed

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

```
Please Select An Operation:
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):
4
Please Enter a Year to Remove it's Record: 2020
Error: No such data with that year on record

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

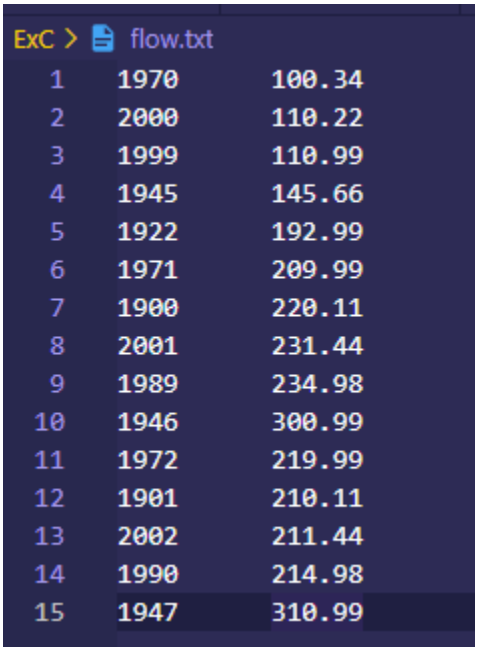
Output after pressing 1 to view the data after the additions and removals

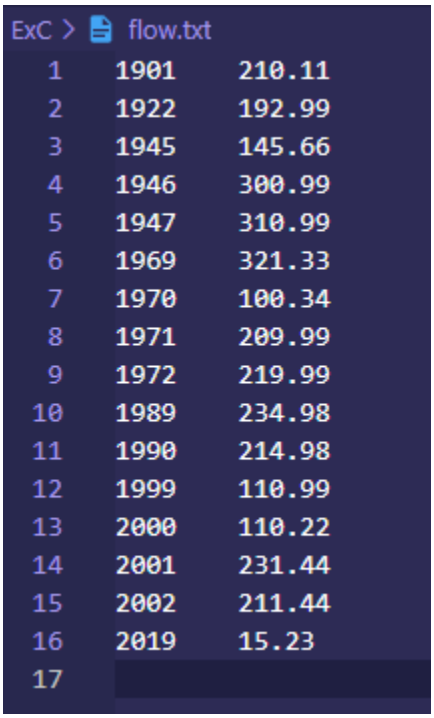
```
Please Select An Operation:
1.      Display Flow List, and the Average
2.      Add Data
3.      Save Data Into the File
4.      Remove Data
5.      Quit
Enter Your Choice (1, 2, 3, 4, or 5):
1
Year    Flow (in billions of cubic meters)
1901    210.11
1922    192.99
1945    145.66
1946    300.99
1947    310.99
1969    321.33
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
2019    15.23

Number of Data Points: 16
The annual average of the flow is: 196.354 billions of cubic meters

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

Output after pressing 3 to save the new data to the flow.txt text file

```
Please Select An Operation:
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):
3
Data is saved into the file flow.txt

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

The text file before and after saving the data





Output after pressing 5 and terminating the program

```
Please Select An Operation:
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):
5

Program Terminated!
```