Course: Programming Fundamental – ENSF 337

Lab #: Lab 8

Instructor: M. Moussavi

Student Name: Michael Jeremy Olea

Lab Section: B01

Date submitted: Nov. 21, 2018

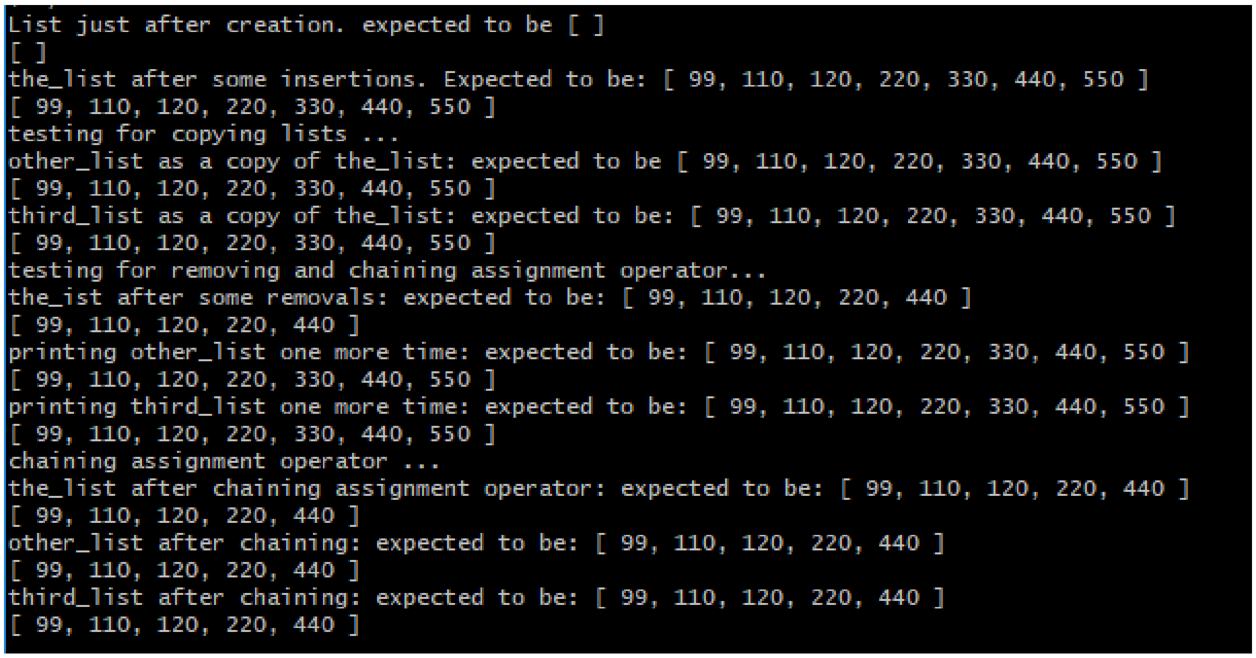
Exercise D;

//OLList.cpp

```cpp
#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->item == itemA)
        {
            before->next = maybe_doomed->next;
            delete maybe_doomed;
        }
        // point three
    }
}

void OLList::destroy()
{
    while(headM != nullptr)
    {
        Node* prev_node = headM;
        headM = headM->next;
        delete prev_node;
    }
    headM = 0;
}

void OLList::copy(const OLList& source)
{
    Node* position = source.headM;
    Node* prev_node = nullptr;
    headM = nullptr;
    while(position != nullptr)
    {
        Node* new_node = new Node;
        new_node->item = position->item;
        new_node->next = nullptr;
        position = position->next;
        if(headM == nullptr)
        {
            headM = new_node;
            prev_node = new_node;
        }
        else
        {
            prev_node->next = new_node;
            prev_node = new_node;
        }
    }
}
```

Output: next page



```
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 E:

//List.h

```
#ifndef list_h
#define list_h

struct ListItem {
    int year;
    double flow;
};

struct Node {
    ListItem item;
    Node *next;
};

class FlowList {
public:
    FlowList();
    void add(int year, double flow);
    void insert(const ListItem& itemA);
    void print() const;
    Node* get_headM() const;
    void remove(int year);
```

```
 private:
        Node *headM;
};

#endif
```

//hydro.h

```
#ifndef hydro_h
#define hydro_h

void displayHeader();

int readData(FlowList &list);

int menu();

void display(FlowList &list, int size);

void addData(FlowList &list, int &numRecords);

void removeData(FlowList &list);

double average(FlowList &list, int size);

double median(FlowList &list, int size);

void saveData(FlowList &list);

void pressEnter();

#endif
```

//list.cpp

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

```cpp
FlowList::FlowList(): headM(0)
{
}


void FlowList::insert(const ListItem& itemA)
{
    Node *new_node = new Node;
    new_node->item = itemA;

    if (headM == 0 || itemA.flow <= headM->item.flow) {
        new_node->next = headM;
        headM = new_node;

    }
    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.flow > after->item.flow) {
            before = after;
            after = after->next;
        }
        new_node->next = after;
        before->next = new_node;
    }
}

void FlowList::print() const
{
    cout << "Year \t Flow (in billions of cubic meters)\n";
    if (headM != 0) {
        cout << headM->item.year << " \t " << headM->item.flow;
        for (const Node *p = headM->next; p != 0; p = p->next)
            cout << "\n" << p->item.year << " \t " << p->item.flow;
    }
    else
    cout << "Linked-list is empty";
}

Node* FlowList::get_headM() const
{
    return headM;
}

void FlowList::remove(const int year)
```

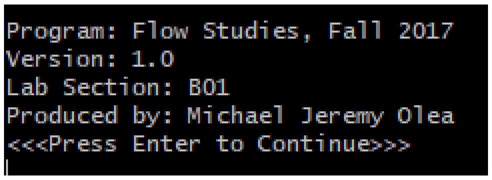
```
{
    if(headM == 0)
        return;
    if(headM->item.year == year)
    {
        Node *doomed = headM;
        headM = headM->next;
        delete doomed;
    }
    else {
        Node *before = headM;
        Node *maybe_doomed = headM->next;
        while(maybe_doomed != 0 && maybe_doomed->item.year != year) {
            before = maybe_doomed;
            maybe_doomed = maybe_doomed->next;
        }
        if(maybe_doomed->item.year == year)
        {
            before->next = maybe_doomed->next;
            delete maybe_doomed;
        }
    }
    return;
}
```

//hydro.cpp

```
#include <iostream>
#include <fstream>
using namespace std;
#define FILENAME "flow.txt"
#include "list.h"
#include "hydro.h"
#include <stdlib.h>


int main(void)
{
    FlowList x;
    int numRecords;
    displayHeader();
    numRecords = readData(x);
    int quit = 0;
    while(1)
    {
        switch(menu())
        {
            case 1:
                display(x, numRecords);
                pressEnter();
                break;
            case 2:
                addData(x, numRecords);
                pressEnter();
                break;
            case 3:
                saveData(x);
                pressEnter();
                break;
            case 4:
                removeData(x);
                pressEnter();
                break;
            case 5:
                cout << "\nProgram terminated!\n\n";
                quit = 1;
                break;
            default:
                cout << "\nNot a valid input.\n";
        }
    if(quit == 1) break;
    }
}

void displayHeader()
{
    cout << "\nProgram: Flow Studies, Fall 2017";
    cout << "\nVersion: 1.0";
    cout << "\nLab Section: B01";
    cout << "\nProduced by: Michael Jeremy Olea";
    pressEnter();
}

void pressEnter()
{
    cout << "\n<<<Press Enter to Continue>>>\n";
    while(1)
    {
        if(cin.get() == '\n')
            return;
    }
}

int menu()
{
    cout << "\nPlease select on the following operations\n";
    cout << "\t1. Display flow list, average and median\n";
    cout << "\t2. Add data.\n";
    cout << "\t3. Save data into the file\n";
    cout << "\t4. Remove data\n";
    cout << "\t5. Quit\n";
    cout << "Enter your choice (1, 2, 3, 4, or 5)\n";
    int selected;
    cin >> selected;
    return selected;
}

int readData(FlowList &list)
{
    int y;
    double f;
    int length = 0;
    ifstream input;
    input.open(FILENAME);
    if(input.fail())
    {
        cout << "\nError: cannot open file 'flow.txt'\n";
        exit(1);
    }
    while(!input.eof())
    {
        input >> y >> f;
        length++;
        ListItem *data = new ListItem {y,f};
        list.insert(*data);
    }
    return length;
}

void display(FlowList &list, int size)
{
    list.print();
    cout << "\n\n";
    cout << "the average is " << average(list, size) << " in billions of cubic
meters\n";
    cout << "the median is " << median(list, size) << " in billions of cubic
meters";
}

double average(FlowList &list, int size)
{
    Node* select = list.get_headM();
    double sum = 0;
    while(select)
    {
        sum += select->item.flow;
        select = select->next;
    }

    return (sum/size);
}

double median(FlowList &list, int size)
{
    Node* med = list.get_headM();
    for(int i = 0; i < (size-1)/2; i++)
    {
        med = med->next;
    }
    return med->item.flow;
}

void addData(FlowList &list, int &numRecords)
{
    Node *check = list.get_headM();
    ListItem *new_item = new ListItem;
    cout << "\nInput a year and a flow\n";
    cin >> new_item->year >> new_item->flow;
    if(cin.fail())
    {
        cout << "\nInvalid input\n";
        return;
    }
    while(check)
    {
        if(check->item.year == new_item->year)
        {
            cout << "\nError: Duplicate year\n";
```

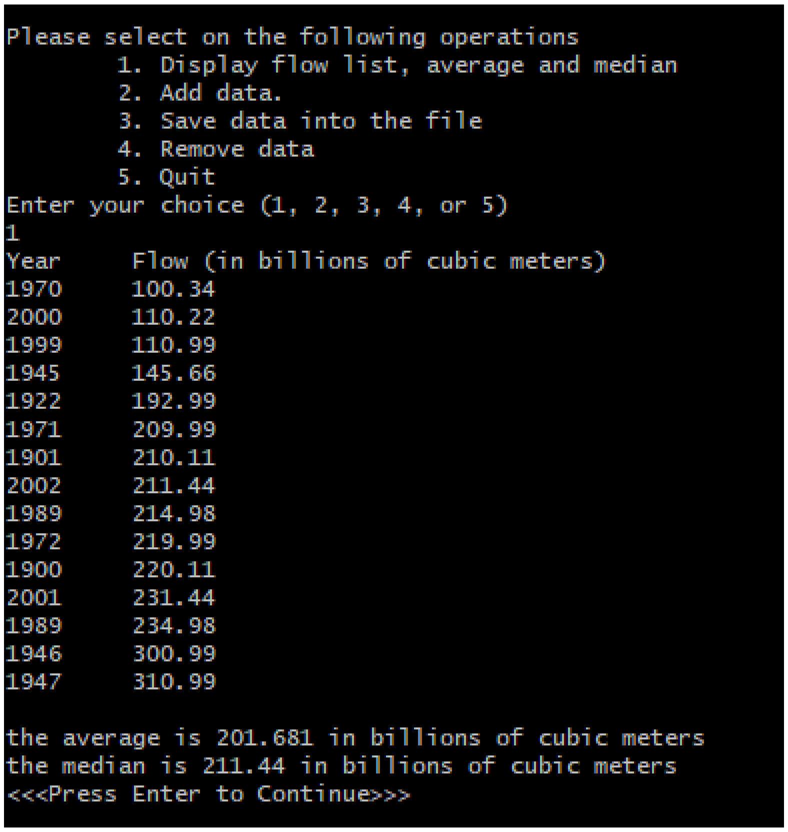
```cpp
            return;
        }
        check = check->next;
    }
    numRecords++;
    list.insert(*new_item);
    cout << "\nData successfully added\n";
}

void saveData(FlowList &list)
{
    ofstream output;
    output.open(FILENAME);
    if(output.fail())
    {
        cout << "\nError: cannot open file 'flow_Output.txt'\n" << endl;
        exit(1);
    }
    Node *select = list.get_headM();
    while(select)
    {
        output << select->item.year << "\t" << select->item.flow;
        output << "\n";
        select = select->next;
    }
    output.close();
    cout << "\nFile successfully saved\n";
}

void removeData(FlowList &list)
{
    Node *check = list.get_headM();
    int removeYear;
    cout << "\nEnter year you want to remove\n";
    cin >> removeYear;
    if(cin.fail())
    {
        cout << "Invalid input";
        return;
    }
    while(check)
    {
        if(check->item.year == removeYear)
        {
            list.remove(removeYear);
            cout << "Year successfully removed";
            return;
        }
        check = check->next;
    }
    cout << "\nYear does not exist\n";
}
```

Output:



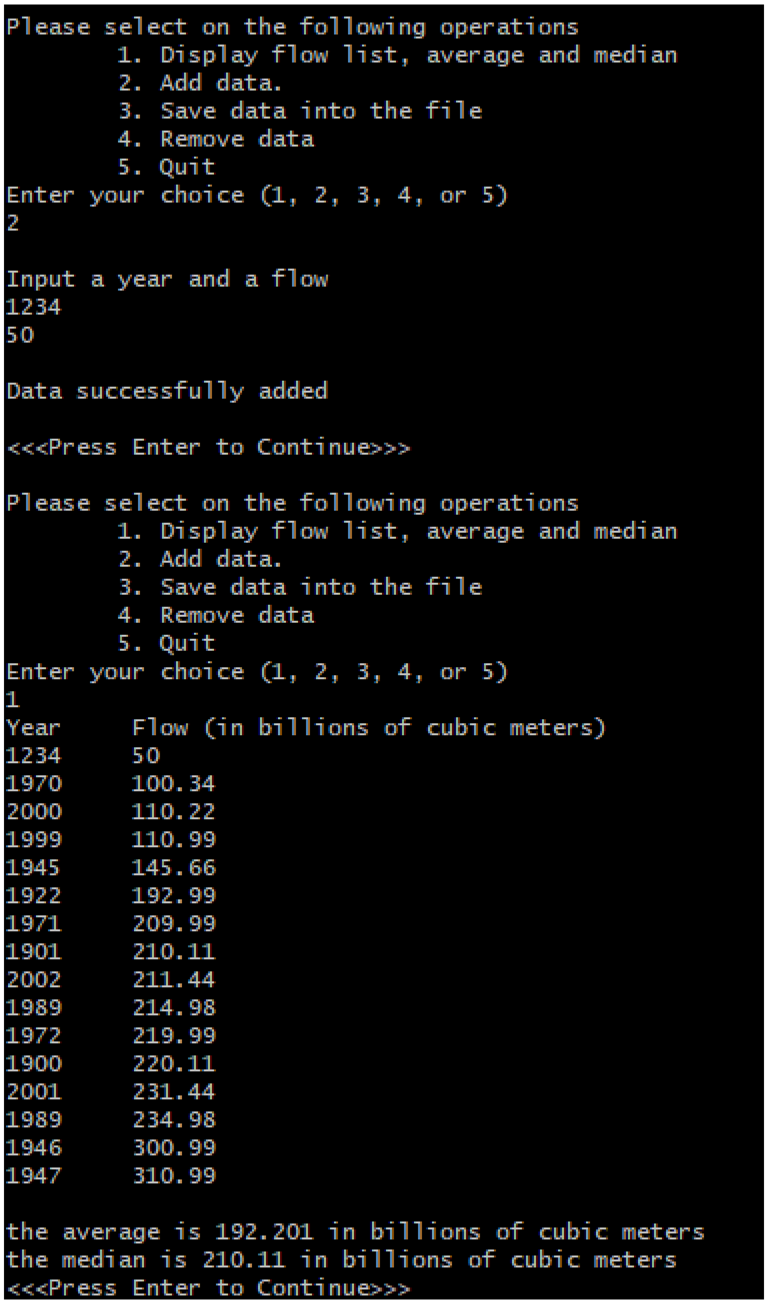
```
Program: Flow Studies, Fall 2017
Version: 1.0
Lab Section: B01
Produced by: Michael Jeremy Olea
<<<Press Enter to Continue>>>
```

Enter 1 to display data



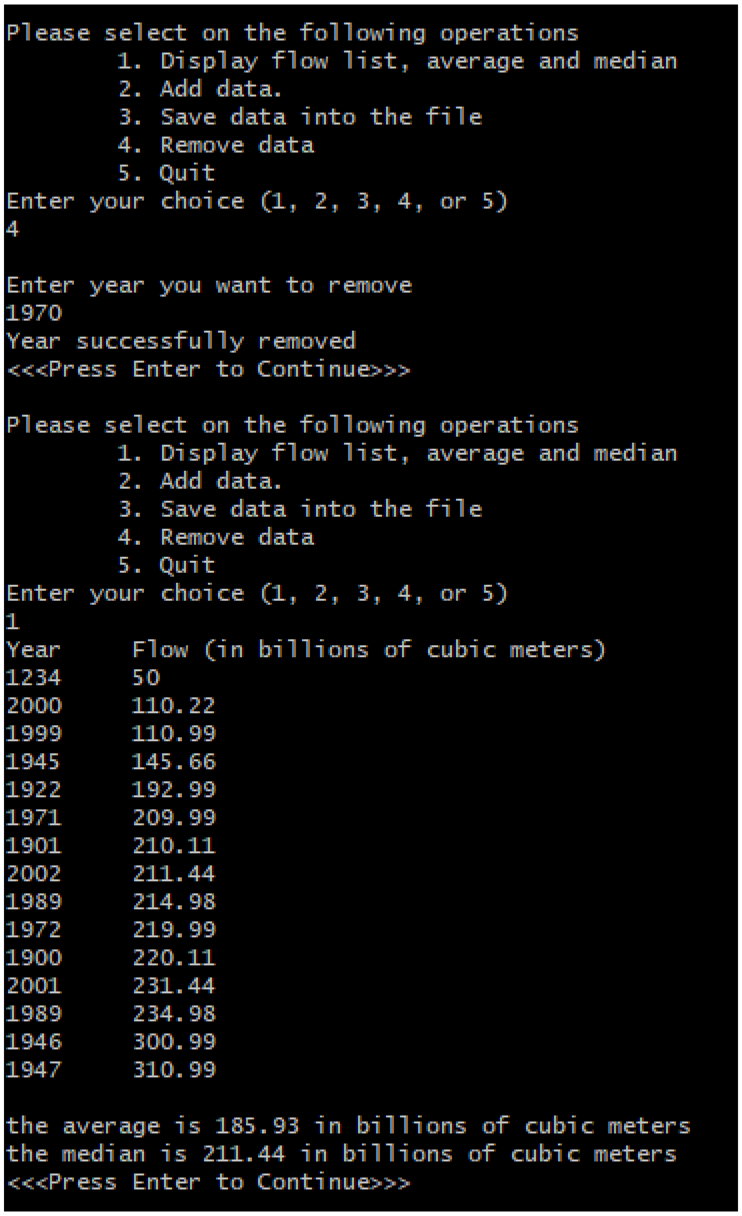
```
Please select on the following operations
        1. Display flow list, average and median
        2. Add data.
        3. Save data into the file
        4. Remove data
        5. Quit
Enter your choice (1, 2, 3, 4, or 5)
1
Year     Flow (in billions of cubic meters)
1970     100.34
2000     110.22
1999     110.99
1945     145.66
1922     192.99
1971     209.99
1901     210.11
2002     211.44
1989     214.98
1972     219.99
1900     220.11
2001     231.44
1989     234.98
1946     300.99
1947     310.99

the average is 201.681 in billions of cubic meters
the median is 211.44 in billions of cubic meters
<<<Press Enter to Continue>>>
```

Enter 2 to add data, input a year and flow and display it to prove it works



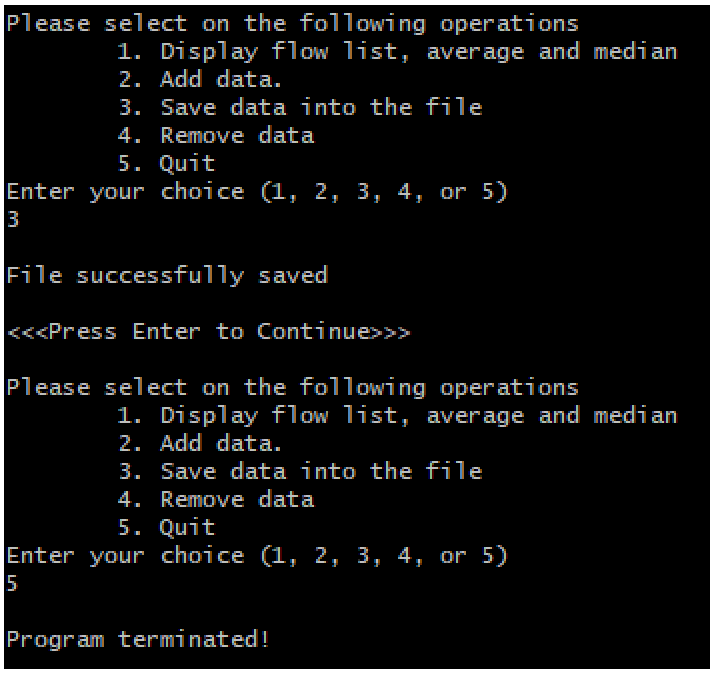
```
Please select on the following operations
        1. Display flow list, average and median
        2. Add data.
        3. Save data into the file
        4. Remove data
        5. Quit
Enter your choice (1, 2, 3, 4, or 5)
2

Input a year and a flow
1234
50

Data successfully added

<<<Press Enter to Continue>>>

Please select on the following operations
        1. Display flow list, average and median
        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)
1234     50
1970     100.34
2000     110.22
1999     110.99
1945     145.66
1922     192.99
1971     209.99
1901     210.11
2002     211.44
1989     214.98
1972     219.99
1900     220.11
2001     231.44
1989     234.98
1946     300.99
1947     310.99

the average is 192.201 in billions of cubic meters
the median is 210.11 in billions of cubic meters
<<<Press Enter to Continue>>>
```

Enter 4 to remove data, remove year “1970” then display to prove it works



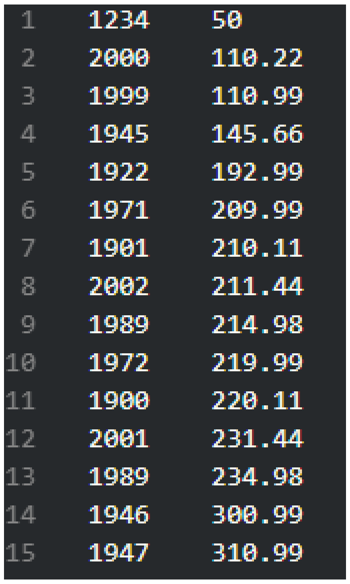
```
Please select on the following operations
        1. Display flow list, average and median
        2. Add data.
        3. Save data into the file
        4. Remove data
        5. Quit
Enter your choice (1, 2, 3, 4, or 5)
4

Enter year you want to remove
1970
Year successfully removed
<<<Press Enter to Continue>>>

Please select on the following operations
        1. Display flow list, average and median
        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)
1234     50
2000     110.22
1999     110.99
1945     145.66
1922     192.99
1971     209.99
1901     210.11
2002     211.44
1989     214.98
1972     219.99
1900     220.11
2001     231.44
1989     234.98
1946     300.99
1947     310.99

the average is 185.93 in billions of cubic meters
the median is 211.44 in billions of cubic meters
<<<Press Enter to Continue>>>
```

Enter 3 to save data, then press 5 to terminate



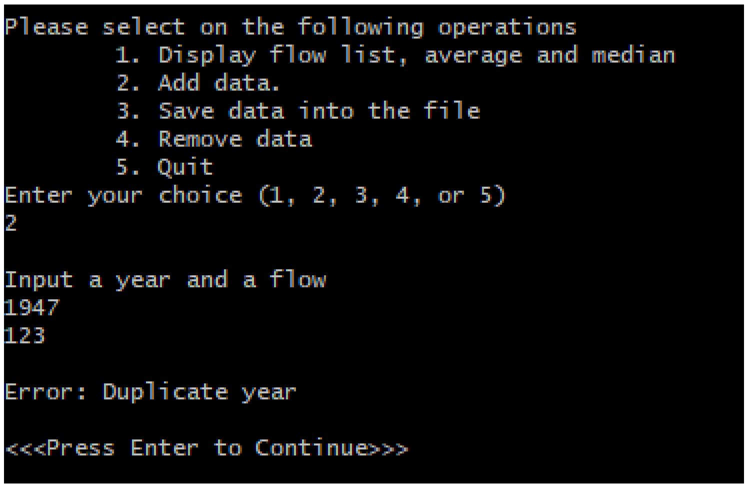
```
Please select on the following operations
        1. Display flow list, average and median
        2. Add data.
        3. Save data into the file
        4. Remove data
        5. Quit
Enter your choice (1, 2, 3, 4, or 5)
3

File successfully saved

<<<Press Enter to Continue>>>

Please select on the following operations
        1. Display flow list, average and median
        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!
```

New “flow.txt” file



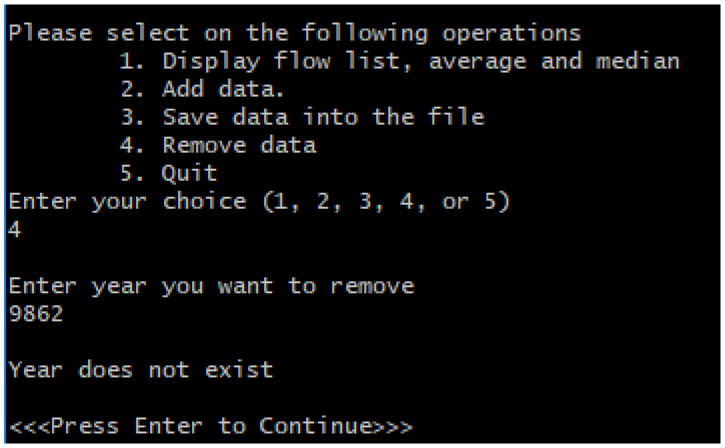
```
1234    50
2000    110.22
1999    110.99
1945    145.66
1922    192.99
1971    209.99
1901    210.11
2002    211.44
1989    214.98
1972    219.99
1900    220.11
2001    231.44
1989    234.98
1946    300.99
1947    310.99
```

Enter 2 to add data and enter duplicate year to get error message



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

Input a year and a flow
1947
123

Error: Duplicate year

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

Enter 3 to remove data and enter non-existent year



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

Enter year you want to remove
9862

Year does not exist

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