Table C55 PSP2 Project Plan Summary

Student	James Small	Date	3/9/14
Program	6A	Program #	9
Instructor	Dr. Concepcion	Language	C++

Summary	**Plan**	**Actual**	**To Date**
LOC/Hour	63.5	38.7	59
Actual Time		121	819
Planned Time	85		680
CPI(Cost-Performance Index)			0.83 (Actual/Planned)
% Reused	36.6	37.6	15.4
% New Reused	0	0	26
Test Defects/KLOC	17.9	12.8	17.4
Total Defects/KLOC	35.8	51.3	37.3
Yield %	15.385	50	20

Program Size (LOC):	**Plan**	**Actual**	**To Date**
Base(B)	278 (Measured)	278 (Measured)	
Deleted (D)	4 (Estimated)	6 (Counted)	
Modified (M)	2 (Estimated)	3 (Counted)	
Added (A)	88 (N-M)	75 (T-B+D-R)	
Reused (R)	209 (Estimated)	209 (Counted)	328
Total New & Changed (N)	90 (Estimated)	78 (A+M)	805
Total LOC (T)	571 (N+B-M-D+R)	556 (Measured)	2127
Total New Reused	0	0	209
Upper Prediction Interval (70%)	233.194		
Lower Prediction Interval (70%)	67.7976		

Time in Phase (min.)	**Plan**	**Actual**	**To Date**	**To Date %**
Planning	4	6	35	4.3
Design	9	10	85	10.4
Design review	1	8	18	2.2
Code	30	39	288	35.2
Code review	2	14	29	3.5
Compile	5	16	54	6.6
Test	23	11	198	24.2
Postmortem	12	17	112	13.7
Total	85	121	819	100
Total Time UPI (70%)	129.15			
Total Time LPI (70%)	72.2603			

(continued)

Table C55 PSP2 Project Plan Summary (continued)

Student	James Small	Date	3/4/14
Program	4A	Program #	8
Instructor	Dr. Concepcion	Language	C++

Defects Injected	***Plan***	**Actual**	**To Date**	**To Date %**
Planning	0	0	0	0
Design	0.1	0	1	3.3
Design review	0	0	0	0
Code	3.1	4	29	96.7
Code review	0	0	0	0
Compile	0	0	0	0
Test	0	0	0	0
Total Development	3.2	4	30	100

Defects Removed	***Plan***	**Actual**	**To Date**	**To Date %**
Planning	0	0	0	0
Design	0	0	0	0
Design review	0	0	0	0
Code	0	0	0	0
Code review	0.5	2	6	20
Compile	1.1	1	10	33.3
Test	1.6	1	14	46.7
Total Development	3.2	4	30	100
After Development	0	0	0	

Defect Removal Efficiency	***Plan***	***Actual***	***To Date***
Defects/Hour - Design review	*0*	*0*	*0*
Defects/Hour - Code review	*16.27*	*8.57*	*12.41*
Defects/Hour - Compile	*14.45*	*3.75*	*11.11*
Defects/Hour - Test	*4.24*	*5.45*	*4.24*
DRL(DLDR/UT)	*0*	*0*	*0*
DRL(CodeReview/UT)	*3.84*	*1.57*	*2.93*
DRL(Compile/UT)	*3.41*	*0.69*	*2.62*

Table C39 Size Estimating Template

Student James Small Date 3/9/14
Instructor Dr. Concepcion Program # 9

BASE PROGRAM LOC				ESTIMATE	ACTUAL
BASE SIZE (B) => => => => => => => => => =>				278	278
LOC DELETED (D) => => => => => => => => =>				4	6
LOC MODIFIED (M) => => => => => => => => =>				2	3
OBJECT LOC					
BASE ADDITIONS	TYPE[1]	METHODS	REL. SIZE	LOC	LOC
TOTAL BASE ADDITIONS (BA) => => => => => => =>					
NEW OBJECTS	TYPE	METHODS	REL. SIZE	LOC (New Reused*)	
Linear Regression	Calc	9	Medium	101	140*
TOTAL NEW OBJECTS (NO) => => => => => => =>				101	140
REUSED OBJECTS					
StringToFloat (3B)				50	50
FileCheck (4B)				19	19
LinearRegression (4A)				140	140
REUSED TOTAL (R) => => => => => => => => =>				209	209
				SIZE	TIME
Estimated Object LOC (E):	$E = BA + NO + M$			2	
Regression Parameters:	β_0 (size and time)			151.055	100.656
Regression Parameters:	β_1 (size and time)			-0.279586	0.024859
Estimated New and Changed LOC (N):	$N = \beta_0 + \beta_1 * E$			150.496	
Estimated Total LOC:	$T = N + B - D - M + R$			628.496	
Estimated Total New Reuse (sum of * LOC):				0	
Estimated Total Development Time:	$Time = \beta_0 + \beta_1 * E$				100.705
Prediction Range:	Range			82.6983	28.445
Upper Prediction Interval:	$UPI = N + Range$			233.194	129.15
Lower Prediction Interval:	$LPI = N - Range$			67.7976	72.2603
Prediction Interval Percent:				70%	70%

[1] L=Logic, I=I/O, C=Calculation, T=Text, D=Data, S=Set-up

Compilation

```
jamess-imac:program AcousticTime$ g++ -c FileCheck.cpp
jamess-imac:program AcousticTime$ g++ -c Input.cpp
jamess-imac:program AcousticTime$ g++ -c LinearRegression.cpp
jamess-imac:program AcousticTime$ g++ -c StringToFloat.cpp
jamess-imac:program AcousticTime$ g++ -o program4A program4A.cpp FileCheck.o
Input.o LinearRegression.o StringToFloat.o
jamess-imac:program AcousticTime$
```

Test 1

```
jamess-imac:program AcousticTime$ ./program6A
What would you like to do?
Enter 1 to read from file.
Enter 2 to write to file.
Enter 3 to modify a file.
Enter 4 to calculate linear regression and prediction interval.
Enter 0 to quit.
Choice: 4
Enter the x-axis values filename: xvalues

Enter the y-axis values filename: yvalues
Enter the estimated object LOC to use: 386

B0 = -22.5524
B1 = 1.72793

Range 70% = 229.972
UPI 70% = 874.401
LPI 70% = 414.458
Range 90% = 386.053
UPI 90% = 1030.48
LPI 90% = 258.376

Prediction for 386 = 644.429
jamess-imac:program AcousticTime$
```

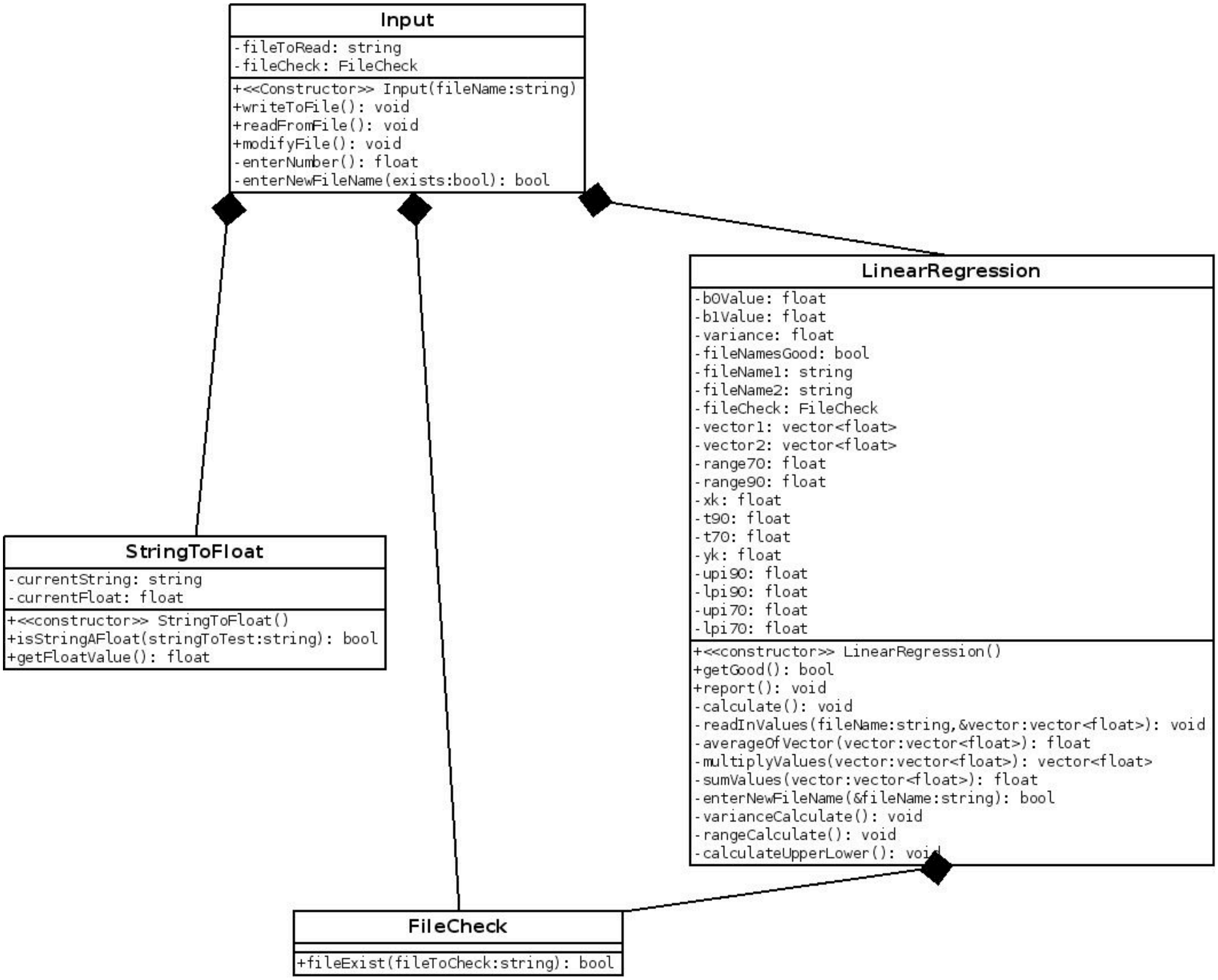
Test 2

```
jamess-imac:program AcousticTime$ ./program6A
What would you like to do?
Enter 1 to read from file.
Enter 2 to write to file.
Enter 3 to modify a file.
Enter 4 to calculate linear regression and prediction interval.
Enter 0 to quit.
Choice: 4
Enter the x-axis values filename: x

Enter the y-axis values filename: y
Enter the estimated object LOC to use: 2

B0 = 151.055
B1 = -0.279586

Range 70% = 82.6983
UPI 70% = 233.194
LPI 70% = 67.7976
Range 90% = 138.826
UPI 90% = 289.322
LPI 90% = 11.6703

Prediction for 2 = 150.496
```

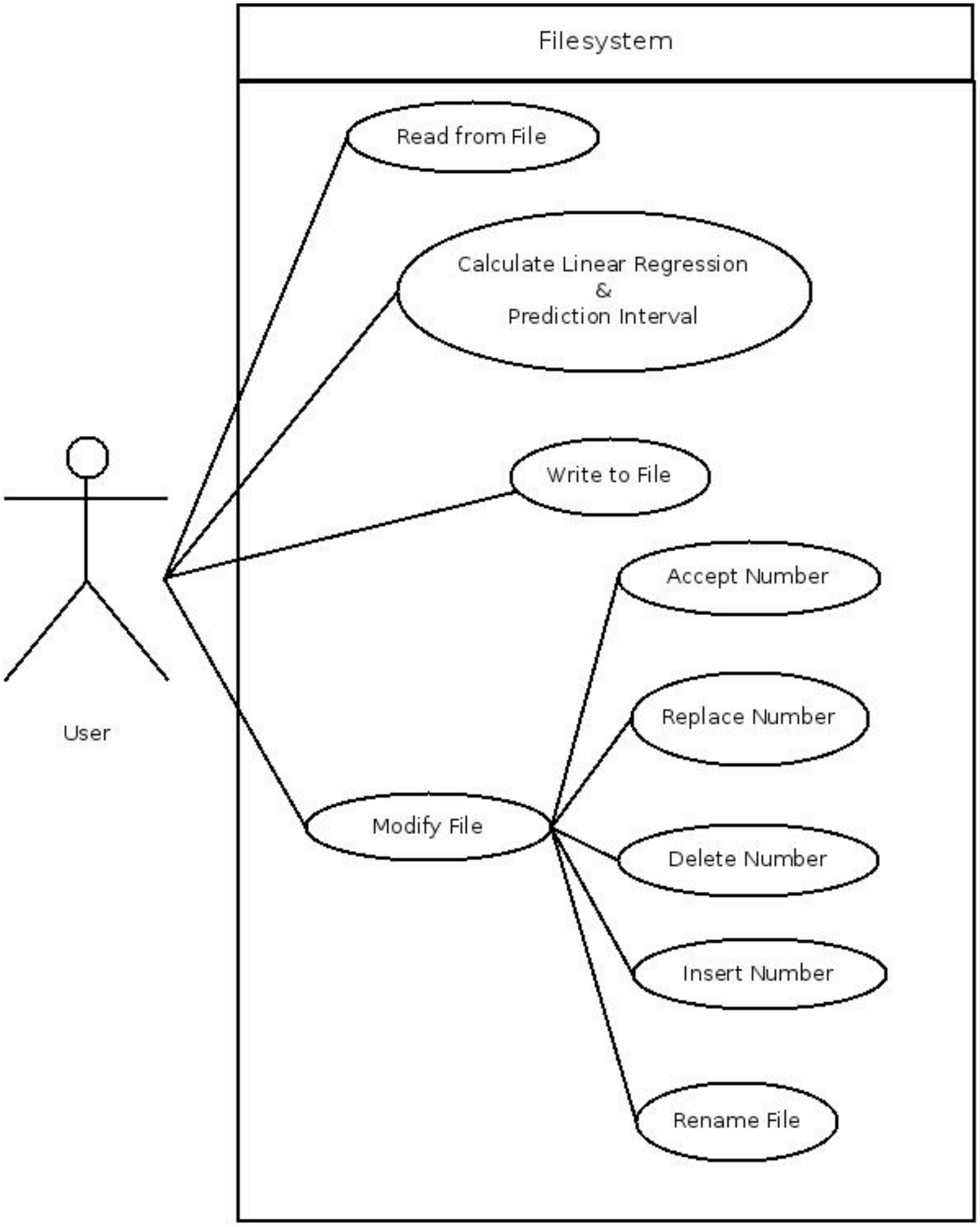
Test 3

```
jamess-imac:program AcousticTime$ ./program6A
What would you like to do?
Enter 1 to read from file.
Enter 2 to write to file.
Enter 3 to modify a file.
Enter 4 to calculate linear regression and prediction interval.
Enter 0 to quit.
Choice: 4
Enter the x-axis values filename: x

Enter the y-axis values filename: y2
Enter the estimated object LOC to use: 2

B0 = 100.656
B1 = 0.024859

Range 70% = 28.445
UPI 70% = 129.15
LPI 70% = 72.2603
Range 90% = 47.7507
UPI 90% = 148.456
LPI 90% = 52.9547

Prediction for 2 = 100.705
```

UML Class Diagram

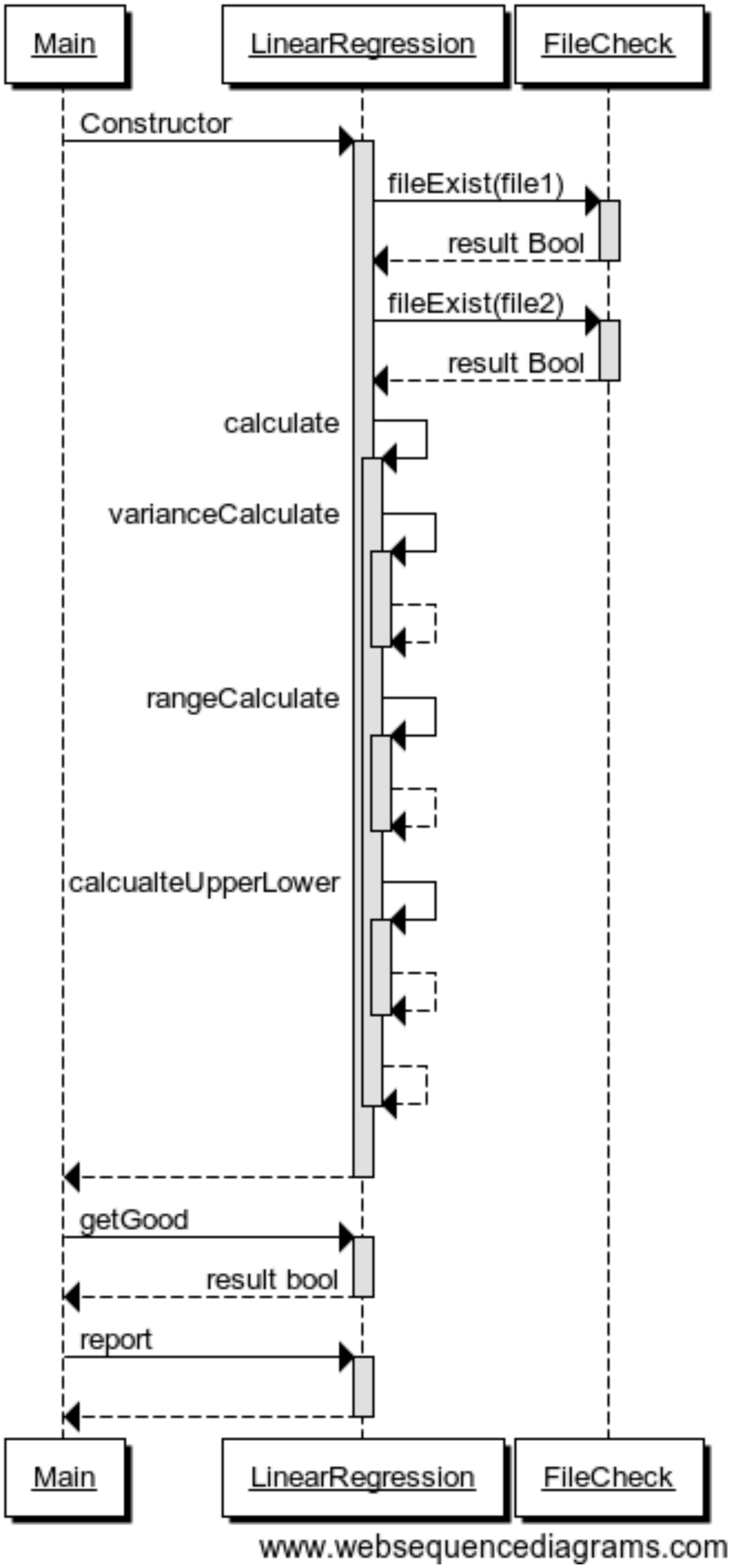


UML Use Case Diagram



Test	**Parameter**	**Expected Value**	**Actual Value**
1	B0	-22.55	-22.5524
	B1	1.7279	1.72793
	UPI 70%	874	874.401
	LPI 70%	414	414.458
	UPI 90%	1030	1030.48
	LPI 90%	258	258.376
2	Estimated New and Changed LOC		150.496
	UPI 70%		233.194
	LPI 70%		67.7976
	UPI 90%		289.322
	LPI 90%		11.6703
	Actual New and Changed LOC		78
3	Estimated New and Changed LOC		150.496
	UPI 70%		129.15
	LPI 70%		72.2603
	UPI 90%		148.456
	LPI 90%		52.9597
	Actual Time		121

Sequence Diagram for Linear Regression and Prediction Interval Scenario



Pseudo-Code for Linear Regression Methods

```
LinearRegression::LinearRegression()

    set default values for variables

    ask user for x-axis file name

    check if x-axis file name is valid choice

    ask user for y-axis file name

    check if y-axis file name is valid choice

    call calculate method


void LinearRegression::calculate()

    b1Value = topValue / bottomValue;
    b0Value = averageOfVector(vector2) - b1Value * averageOfVector(vector1);


    call readInValues for both files

    declare and initialize all need variables to hold temp values

    calculate topValueLeft using sumvalues and multiplyvalues methods
    calcualte topValueRight using averageOfVector method
    calcualte topValue using topValueLeft - topValueRight

    calculate bottomValueLeft using sumvalues and multiplyvalus methods
    calculate bottomValueRight using averageOfvector method
    calculate bottomValue using bottomValueLeft - bottomValueRight

    calculate b1 value using topValue / bottomValue
    calculate b0 value using averageofvector method and b1value


void LinearRegression::readInValues(string filename, vector<float> &vector)

    delcare ifstream variable

    open file

    declare currentValue float and set to 0
    while (lines in file)
        read in value
        add to vector

    close file


vector<float> LinearRegression::multiplyValues(vector<float> vector1,
vector<float> vector2)

    declare vector to hold results

    if (vectors not same size)
        return

    for (all items in vector1)
        add to new vector: vector1[i] * vector2[i]

    return new vector


float LinearRegression::sumValues(vector<float> vector)

    declare variable to hold result

    for (all items in vector)
        add vector[i] to sum

    return sum


void LinearRegression::varianceCalculate()

    calculate sum of yi - b0 - b1 * xi
    calculate variance
        sort of (1 / (n - 2)) * sum from above


void LinearRegression::rangeCalculate()

    calculate range
        calculate top right value
            (xk - average(x values)) * (xk - average(x values))
        calculate bottom right value
            sum of (xi - average(x values)) * (xi - average(x values))
        calculate right value
            top right value / bottom right value
        calculate range
           sqrt of  1 + (1 / n) + right value
           above ∗ variance
           range 90 = range ∗ 90% interval
           range 70 = range ∗ 70% interval

void LinearRegression::calculateUpperLower()

    calculate yk
        b0 + b1 ∗ xk
    calculate upi and lpi
        yk + range70
        yk − range70
        yk + range90
        yk − range90
```

Object Category Sizes in LOC per Method

Name: James Small

Program: 6A Number: 8

Instructor: Dr. Concepcion Language: C++

Object Size in LOC per Method (stddev method)

Type	V. Small	Small	Medium	Large	V. Large
Logic	0	0	0	0	0
I/O	11.26	19.23	27.2	35.17	43.14
Calc	15.56	15.56	15.56	15.56	15.56
Text	0	0	0	0	0
Data	16.67	16.67	16.67	16.67	16.67
Set-up	0	0	0	0	0

Object Size in LOC per Method (natural log method)

Type	V. Small	Small	Medium	Large	V. Large
Logic	0	0	0	0	0
I/O	14.73	19.6	26.09	34.73	46.23
Calc	15.56	15.56	15.56	15.56	15.56
Text	0	0	0	0	0
Data	16.67	16.67	16.67	16.67	16.67
Set-up	0	0	0	0	0

```
// Name: James Small
// Program: 4B
// Class: CSE455
// Description: Program to input, output, or modify, and calculate linear
    regression.

#include <iostream>
#include <string>
#include <stdlib.h> // for atoi
#include <ctype.h> // for isdigit
#include "Input.h"
#include "LinearRegression.h"

using namespace std;

int main()
{
    char choice = 0;
    bool choiceGood = false;

    do {
        cout << "What would you like to do?\n";
        cout << "Enter 1 to read from file.\n";
        cout << "Enter 2 to write to file.\n";
        cout << "Enter 3 to modify a file.\n";
        cout << "Enter 4 to calculate linear regression and prediction interval.
            \n";
        cout << "Enter 0 to quit.\n";
        cout << "Choice: ";

        cin >> choice;

        if (isdigit(choice)) {
            if (atoi(&choice) >= 0 && atoi(&choice) < 5)
                choiceGood = true;
            else
                cout << "\nInvalid Choice, Try again\n\n";
        } else
            cout << "\nInvalid Choice, Try again\n\n";

        cin.ignore(INT_MAX,'\n');

    } while (!choiceGood);

    if (choice != '0') {

        if (choice == '1') {
            Input input;
            input.readFromFile();
        }
        else if (choice == '2') {
            Input input;
            input.writeToFile();
        }
        else if (choice == '3') {
            Input input;
            input.modifyFile();
        }
        else if (choice == '4') {
            LinearRegression linear;

            if (linear.getGood())
                linear.report();
        }
    }

    return 0;
}
```

```
// Name: James Small
// Program: 4B
// Class: CSE455
// Description: Class to check if file exists in current
directory

#ifndef FILECHECK_H
#define FILECHECK_H

#include <string>

using namespace std;

class FileCheck
{
     public:
           bool fileExist(string fileToCheck);
};
#endif
```

```
// Name: James Small
// Program: 4B
// Class: CSE455
// Description: FileCheck class implementation file

#include "FileCheck.h"
#include <fstream>

// This method takes a string and returns true or false if a
float

bool FileCheck::fileExist(string fileToCheck)
{
    ifstream infile;

    infile.open(fileToCheck.c_str());

    infile.close();

    return infile;
}
```

```
// Name: James Small
// Program: 3B
// Class: CSE455
// Description: Input class Header File

#ifndef INPUT_H
#define INPUT_H

#include <string>
#include "StringToFloat.h"
#include "FileCheck.h"

using namespace std;

class Input
{
public:
    Input();
    void writeToFile();
    void readFromFile();
    void modifyFile();

private :
    string fileToRead;
    float enterNumber();
    bool enterNewFileName(bool exists);
    StringToFloat stringToFloat;
    FileCheck fileCheck;
};
#endif
```

```
// Name: James Small
// Program: 3B
// Class: CSE455
// Description: Input class Implementation File

#include "Input.h"
#include <fstream>
#include <iostream>
#include <vector>
#include <stdlib.h> // for atoi
#include <ctype.h> // for isdigit

using namespace std;

// This is the default constructor

Input::Input()
{
    cout << "Enter the file name to access: ";
    cin >> fileToRead;
}

// This method asks user for a set of numbers and outputs them to
a file

void Input::writeToFile()
{
    while (fileCheck.fileExist(fileToRead))
        if (!enterNewFileName(true))
            return;

    string count;
    float currentValue;
    string currentString = "";
    bool countGood = false;

    do {
        cout << "Enter the amount of numbers to write: ";

        cin >> count;

        bool allDigitsInt = true;

        for (int i = 0; i < count.size(); i++)
            if (!isdigit(count[i]))
                allDigitsInt = false;

        if (allDigitsInt) {
            if (atoi(count.c_str()) > 0)
                countGood = true;
            else
                cout << "\nInvalid number, Try again\n\n";
        } else
            cout << "\nInvalid number, Try again\n\n";

        cin.ignore(INT_MAX,'\n');

    } while (!countGood);

    ofstream outfile;

    outfile.open(fileToRead.c_str());

    for (int i = 0; i < atoi(count.c_str()); i++) {

        cout << "Enter number " << i + 1 << ": ";

        cin >> currentString;

        while (!stringToFloat.isStringAFloat(currentString)) {

            cout << "\nInvalid Value, try again\n\n";
            cout << "Enter number " << i + 1 << ": ";

            cin.ignore(INT_MAX,'\n');

            cin >> currentString;
        }

        currentValue = stringToFloat.getFloatValue();

        if (i == atoi(count.c_str()) - 1)
            outfile << currentValue;
        else
            outfile << currentValue << " ";
    }

    outfile.close();
}

// This method reads in a set of numbers from a file and displays
them on screen

void Input::readFromFile()
{
    while (!fileCheck.fileExist(fileToRead))
        if (!enterNewFileName(false))
            return;

    ifstream infile;

    infile.open(fileToRead.c_str());

    float currentValue = 0;
    while (!infile.eof()) {
        infile >> currentValue;
        cout << currentValue << endl;
    }

    infile.close();
}

// This method modifies an existing file one line at a time.

void Input::modifyFile()
{
    while (!fileCheck.fileExist(fileToRead))
        if (!enterNewFileName(false))
            return;

    ifstream infile;

    infile.open(fileToRead.c_str());

    float currentValue = 0;
    char choice;
    vector<float> currentNumbers;
    bool acceptAllNumbers = false;

    while (!infile.eof()) {
        infile >> currentValue;

        if (acceptAllNumbers) {
            currentNumbers.push_back(currentValue);
        } else {

            bool choiceGood = false;
            do {
                cout << "\nWhat would you like to do with this
number, " << currentValue << "?\n";
                cout << "Enter 1 to accept this number.\n";
                cout << "Enter 2 to replace this number.\n";
                cout << "Enter 3 to delete this number.\n";
                cout << "Enter 4 to insert a new number after
current number.\n";
                cout << "Enter 5 to accept the remainder of the
numbers.\n";
                cout << "Choice: ";

                cin >> choice;

                if (isdigit(choice)) {
                    if (atoi(&choice) > 0 && atoi(&choice) < 6)
                        choiceGood = true;
                    else
```

```
                        cout << "\nInvalid Choice, Try again\n
\n";
                } else
                    cout << "\nInvalid Choice, Try again\n\n";

                cin.ignore(INT_MAX,'\n');

            } while (!choiceGood);

            switch (choice) {
                case '1':
                    currentNumbers.push_back(currentValue);
                    break;
                case '2':
                    currentNumbers.push_back(enterNumber());
                    break;
                case '3':
                    break;
                case '4':
                    currentNumbers.push_back(currentValue);
                    currentNumbers.push_back(enterNumber());
                    break;
                case '5':
                    currentNumbers.push_back(currentValue);
                    acceptAllNumbers = true;
                    break;
                default:
                    break;
            }
        }
    }

    infile.close();

    bool choiceGood = false;

    do {
        cout << "\nWould you like to replace the current file or
create a new file?\n";
        cout << "Enter 1 to replace the current file's contents.
\n";
        cout << "Enter 2 to create a new file.\n";
        cout << "Choice: ";

        cin >> choice;

        if (isdigit(choice)) {
            if (atoi(&choice) > 0 && atoi(&choice) < 3)
                choiceGood = true;
            else
                cout << "\nInvalid Choice, Try again\n\n";
        } else
```

```
            cout << "\nInvalid Choice, Try again\n\n";

        cin.ignore(INT_MAX,'\n');

    } while (!choiceGood);

    if (choice == '2') {
        cout << "Enter the file name to access: ";
        cin >> fileToRead;

        while (fileCheck.fileExist(fileToRead))
            if (!enterNewFileName(false))
                return;
    }

    ofstream outfile;

    outfile.open(fileToRead.c_str());

    for (int i = 0; i < currentNumbers.size(); i++) {
        if (i == currentNumbers.size() - 1)
            outfile << currentNumbers[i];
        else
            outfile << currentNumbers[i] << " ";
    }
}

// This method allows input of a float

float Input::enterNumber()
{
    float current = 0;
    string currentString = "";

    cout << "\nEnter number: ";

    cin >> currentString;

    while (!stringToFloat.isStringAFloat(currentString)) {

        cout << "\nInvalid Value, try again\n\n";
        cout << "\nEnter number: ";

        cin >> currentString;
    }

    current = stringToFloat.getFloatValue();

    return current;
}

// This method asks the user to enter a new filename
```

```
bool Input::enterNewFileName(bool exists)
{
    if (exists)
        cout << "\nThe filename already exists\n";
    else
        cout << "\nThe filename doesn't exist\n";

    char choice = 0;
    bool choiceGood = false;

    do {
        cout << "What would you like to enter a new filename?\n";
        cout << "Enter 1 to enter another filename.\n";
        cout << "Enter 0 to quit.\n";
        cout << "Choice: ";

        cin >> choice;

        if (isdigit(choice)) {
            if (atoi(&choice) >= 0 && atoi(&choice) < 2)
                choiceGood = true;
            else
                cout << "\nInvalid Choice, Try again\n\n";
        } else
            cout << "\nInvalid Choice, Try again\n\n";

        cin.ignore(INT_MAX,'\n');

    } while (!choiceGood);

    if (choice == '1') {
        cout << "Enter the file name to access: ";
        cin >> this->fileToRead;
        return true;
    } else
        return false;
}
```

```
// Name: James Small
// Program: 3B
// Class: CSE455
// Description: Class to convert string to float, if possible

#ifndef STRINGTOFLOAT_H
#define STRINGTOFLOAT_H

#include <string>

using namespace std;

class StringToFloat
{
	public:
		StringToFloat();
		bool isStringAFloat(string stringToTest);
		float getFloatValue();

	private:
		string currentString;
		float currentFloat;
};
#endif
```

```
// Name: James Small
// Program: 3B
// Class: CSE455
// Description: StringToFloat class implementation file

#include "StringToFloat.h"
#include <stdlib.h> // for atof
#include <ctype.h> // for isdigit

// Constructor which sets the currentFloat to 0

StringToFloat::StringToFloat()
{
     currentFloat = 0;
}

// This method takes a string and returns true or false if a
float

bool StringToFloat::isStringAFloat(string stringToTest)
{
     currentString = stringToTest;
     int periodsCount = 0;
     bool nonDigitFound = false;
     bool isFloat = false;

     for (int i = 0;i < currentString.length(); i++) {
          if (!isdigit(currentString[i])) {
               if (currentString[i] == '.') {
                    periodsCount++;
               }
               else if (currentString[i] == '-') {
                    if (i != 0)
                         nonDigitFound = true;
               } else
                    nonDigitFound = true;
          }
     }

     if (!nonDigitFound && periodsCount < 2) {
          isFloat = true;
          currentFloat = atof(currentString.c_str());
     }

     return isFloat;
}

// This method returns the float value

float StringToFloat::getFloatValue()
{
     return currentFloat;
}
```

```
// Name: James Small
// Program: 4a
// Class: CSE455
// Description: Class to calculate the linear regression of a set of numbers

#ifndef LINEARREGRESSION_H
#define LINEARREGRESSION_H

#include <string>
#include <vector>
#include "FileCheck.h"

using namespace std;

class LinearRegression
{
    public:
        LinearRegression();
        bool getGood();
        void report();

    private:
        float b0Value;
        float b1Value;
        float variance;
        float range70;
        float range90;
        float xk;
        float t90;
        float t70;
        float yk;
        float upi90;
        float lpi90;
        float upi70;
        float lpi70;
        bool fileNamesGood;
        string fileName1;
        string fileName2;
        FileCheck fileCheck;
        vector<float> vector1;
        vector<float> vector2;
        void calculate();
        void readInValues(string fileName, vector<float> &vector);
        float averageOfVector(vector<float> vector);
        vector<float> multiplyValues(vector<float> vector1, vector<float> vector2
            );
        float sumValues(vector<float> vector);
        bool enterNewFileName(string &fileName);
        void varianceCalculate();
        void rangeCalculate();
        void calculateUpperLower();
};
#endif
```

```
// Name: James Small
// Program: 4A
// Class: CSE455
// Description: LinearRegression class implementation file

#include "LinearRegression.h"
#include <fstream>
#include <iostream>
#include <math.h>       /* sqrt */

// Constructor that takes in both file names

LinearRegression::LinearRegression()
{
    fileNamesGood = true;
    b0Value = 0;
    b1Value = 0;
    variance = 0;
    range90 = 0;
    range70 = 0;
    t90 = 1.860;
    t70 = 1.108;
    yk = 0;
    lpi90 = 0;
    upi90 = 0;
    lpi70 = 0;
    upi70 = 0;

    cout << "Enter the x-axis values filename: ";
    cin >> fileName1;

    while (!fileCheck.fileExist(fileName1))
        if (!enterNewFileName(fileName1)) {
            fileNamesGood = false;
            return;
        }

    cout << "\nEnter the y-axis values filename: ";
    cin >> fileName2;

    while (!fileCheck.fileExist(fileName2))
        if (!enterNewFileName(fileName2)) {
            fileNamesGood = false;
            return;
        }

    cout << "Enter the estimated object LOC to use: ";
    cin >> xk;

    calculate();
}

// This method returns true if the file names were good

bool LinearRegression::getGood()
{
    return fileNamesGood;
}

// This method calculates the linear regression

void LinearRegression::calculate()
{
    readInValues(fileName1, vector1);
    readInValues(fileName2, vector2);

    float topValue = 0;
    float bottomValue = 0;
    float topValueLeft = 0;
    float topValueRight = 0;
    float bottomValueLeft = 0;
    float bottomValueRight = 0;

    topValueLeft = sumValues(multiplyValues(vector1,vector2));
    topValueRight = vector1.size() * averageOfVector(vector1) * averageOfVector
        (vector2);
    topValue = topValueLeft - topValueRight;

    bottomValueLeft = sumValues(multiplyValues(vector1,vector1));
    bottomValueRight = vector1.size() * averageOfVector(vector1) *
        averageOfVector(vector1);
    bottomValue = bottomValueLeft - bottomValueRight;

    b1Value = topValue / bottomValue;
    b0Value = averageOfVector(vector2) - b1Value * averageOfVector(vector1);

    varianceCalculate();
    rangeCalculate();
    calculateUpperLower();
}

// This method reads the numbers from a file into a vector

void LinearRegression::readInValues(string filename, vector<float> &vector)
{
    ifstream infile;

    infile.open(filename.c_str());

    float currentValue = 0;

    while (!infile.eof()) {
        infile >> currentValue;
        vector.push_back(currentValue);
    }

    infile.close();
}

// This method calcualtes the average value of the vector

float LinearRegression::averageOfVector(vector<float> vector)
```

```cpp
{
    return sumValues(vector) / vector.size();
}

// This method multiples parallel vectors and returns a vector as result

vector<float> LinearRegression::multiplyValues(vector<float> vector1, vector<
    float> vector2)
{
    vector<float> multiplyVector;

    if (vector1.size() !=  vector2.size())
        return multiplyVector;

    for (int i = 0; i < vector1.size(); i++)
        multiplyVector.push_back(vector1[i] * vector2[i]);

    return multiplyVector;
}

// This method displays a report of the results

void LinearRegression::report()
{
    cout << "\nB0 = " << b0Value << endl;
    cout << "B1 = " << b1Value << endl << endl;
    cout << "Range 70% = " << range70 << endl;
    cout << "UPI 70% = " << upi70 << endl;
    cout << "LPI 70% = " << lpi70 << endl;
    cout << "Range 90% = " << range90 << endl;
    cout << "UPI 90% = " << upi90 << endl;
    cout << "LPI 90% = " << lpi90 << endl;
    cout << "\nPrediction for " << xk << " = " << yk << endl;
}

// This method sums all values in the vector

float LinearRegression::sumValues(vector<float> vector)
{
    float sum = 0;

    for (int i = 0; i < vector.size(); i++)
        sum += vector[i];

    return sum;
}

// This method asks the user to enter a new filename

bool LinearRegression::enterNewFileName(string &fileName)
{
    cout << "\nThe filename doesn't exist\n";

    char choice = 0;
    bool choiceGood = false;
    do {
        cout << "What would you like to enter a new filename?\n";
        cout << "Enter 1 to enter another filename.\n";
        cout << "Enter 0 to quit.\n";
        cout << "Choice: ";

        cin >> choice;

        if (isdigit(choice)) {
            if (atoi(&choice) >= 0 && atoi(&choice) < 2)
                choiceGood = true;
            else
                cout << "\nInvalid Choice, Try again\n\n";
        } else
            cout << "\nInvalid Choice, Try again\n\n";

        cin.ignore(INT_MAX,'\n');

    } while (!choiceGood);

    if (choice == '1') {
        cout << "Enter the file name to access: ";
        cin >> fileName;
        return true;
    } else
        return false;
}

// This method calcualtes the variance

void LinearRegression::varianceCalculate()
{
    float currentSum = 0;
    float currentValue = 0;

    for (int i = 0; i < vector1.size(); i++) {
        currentValue = vector2[i] - b0Value - b1Value * vector1[i];
        currentSum += currentValue * currentValue;
    }

    currentValue = 1 / ((float)vector1.size() - 2) * currentSum;
    variance = sqrt(currentValue);
}

// This method calcualtes the range

void LinearRegression::rangeCalculate()
{
    float topValueRight = (xk - averageOfVector(vector1)) * (xk - averageOfVector
        (vector1));

    float currentValue = 0;
    float currentSum = 0;

    for (int i = 0; i < vector1.size();i ++) {
        currentValue = vector1[i] - averageOfVector(vector1);
```

```
            currentSum += currentValue * currentValue;
        }

        float valueRight = topValueRight / currentSum;

        float range = 1 + 1 / (float)vector1.size() + valueRight;

        range = sqrt(range);
        range *= variance;
        range90 = range * t90;
        range70 = range * t70;
    }

    // This method calculates upi and lpi

    void LinearRegression::calculateUpperLower()
    {
        yk = b0Value + b1Value * xk;
        upi70 = yk + range70;
        lpi70 = yk - range70;
        upi90 = yk + range90;
        lpi90 = yk - range90;
    }
```