Table C55 PSP2 Project Plan Summary

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

Summary	**Plan**	**Actual**	**To Date**
LOC/Hour	68.6	84.3	62.5
Actual Time		111	698
Planned Time	105		575
CPI(Cost-Performance Index)			0.824 (Actual/Planned)
% Reused	15.2	14.2	7.6
% New Reused	84.2	89.7	28.7
Test Defects/KLOC	21	6.4	17.9
Total Defects/KLOC	35	38.5	35.8
Yield %	0	66.667	15.385

Program Size (LOC):	**Plan**	**Actual**	**To Date**
Base(B)	266	266	
	(Measured)	(Measured)	
Deleted (D)	0	0	
	(Estimated)	(Counted)	
Modified (M)	0	4	
	(Estimated)	(Counted)	
Added (A)	120	152	
	(N-M)	(T-B+D-R)	
Reused (R)	69	69	119
	(Estimated)	(Counted)	
Total New & Changed (N)	120	156	727
	(Estimated)	(A+M)	
Total LOC (T)	455	487	1571
	(N+B-M-D+R)	(Measured)	
Total New Reused	101	140	209
Upper Prediction Interval (70%)	127.232		
Lower Prediction Interval (70%)	87.232		

Time in Phase (min.)	**Plan**	**Actual**	**To Date**	**To Date %**
Planning	4	5	29	4.2
Design	9	16	75	10.7
Design review	9	10	10	1.4
Code	32	32	249	35.7
Code review	9	15	15	2.1
Compile	5	1	38	5.4
Test	25	15	187	26.8
Postmortem	12	17	95	13.6
Total	105	111	698	100
Total Time UPI (70%)	119.487			
Total Time LPI (70%)	79.487			

(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	0.3	0	1	3.8
Design review	0	0	0	0
Code	6.3	6	25	96.2
Code review	0	0	0	0
Compile	0	0	0	0
Test	0	0	0	0
Total Development	6.7	6	26	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	4	4	15.4
Compile	2.7	1	9	34.6
Test	4	1	13	50
Total Development	6.7	6	26	100
After Development	0	0	0	

Defect Removal Efficiency	***Plan***	***Actual***	***To Date***
Defects/Hour - Design review	*0*	*0*	*0*
Defects/Hour - Code review	*0*	*16*	*16*
Defects/Hour - Compile	*29.25*	*60*	*14.21*
Defects/Hour - Test	*9.44*	*4*	*4.17*
DRL(DLDR/UT)	*0*	*0*	*0*
DRL(CodeReview/UT)	*0*	*4*	*3.84*
DRL(Compile/UT)	*3.1*	*15*	*3.41*

Table C39 Size Estimating Template

Student	James Small	Date	3/4/14
Instructor	Dr. Concepcion	Program #	8

				ESTIMATE	ACTUAL
BASE PROGRAM LOC					
BASE SIZE (B) => => => => => => => => => =>				266	266
LOC DELETED (D) => => => => => => => => =>				0	0
LOC MODIFIED (M) => => => => => => => => =>				0	4
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
REUSED TOTAL (R) => => => => => => => => =>				69	69

		SIZE	TIME
Estimated Object LOC (E):	E = BA + NO + M	101	
Regression Parameters:	β_0 (size and time)	155.929	101.806
Regression Parameters:	β_1 (size and time)	-0.482143	-0.0229592
Estimated New and Changed LOC (N):	$N = \beta_0 + \beta_1 * E$	107.232	
Estimated Total LOC:	T = N + B – D – M + R	438.232	
Estimated Total New Reuse (sum of * LOC):		101	
Estimated Total Development Time:	$\text{Time} = \beta_0 + \beta_1 * E$		99.487
Prediction Range:	Range	20	20
Upper Prediction Interval:	UPI = N + Range	127.232	119.487
Lower Prediction Interval:	LPI = N – Range	87.232	79.487
Prediction Interval Percent:		N/A	N/A

[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$ ./program4A
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.
Enter 0 to quit.
Choice: 4
Enter the x-axis values filename: xvalues

Enter the y-axis values filename: yvalues

B0 = -22.5524
B1 = 1.72793
```

Test 2

```
jamess-imac:program AcousticTime$ ./program4A
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.
Enter 0 to quit.
Choice: 4
Enter the x-axis values filename: xvaluesb

Enter the y-axis values filename: yvalues

B0 = -23.9238
B1 = 1.43097
```

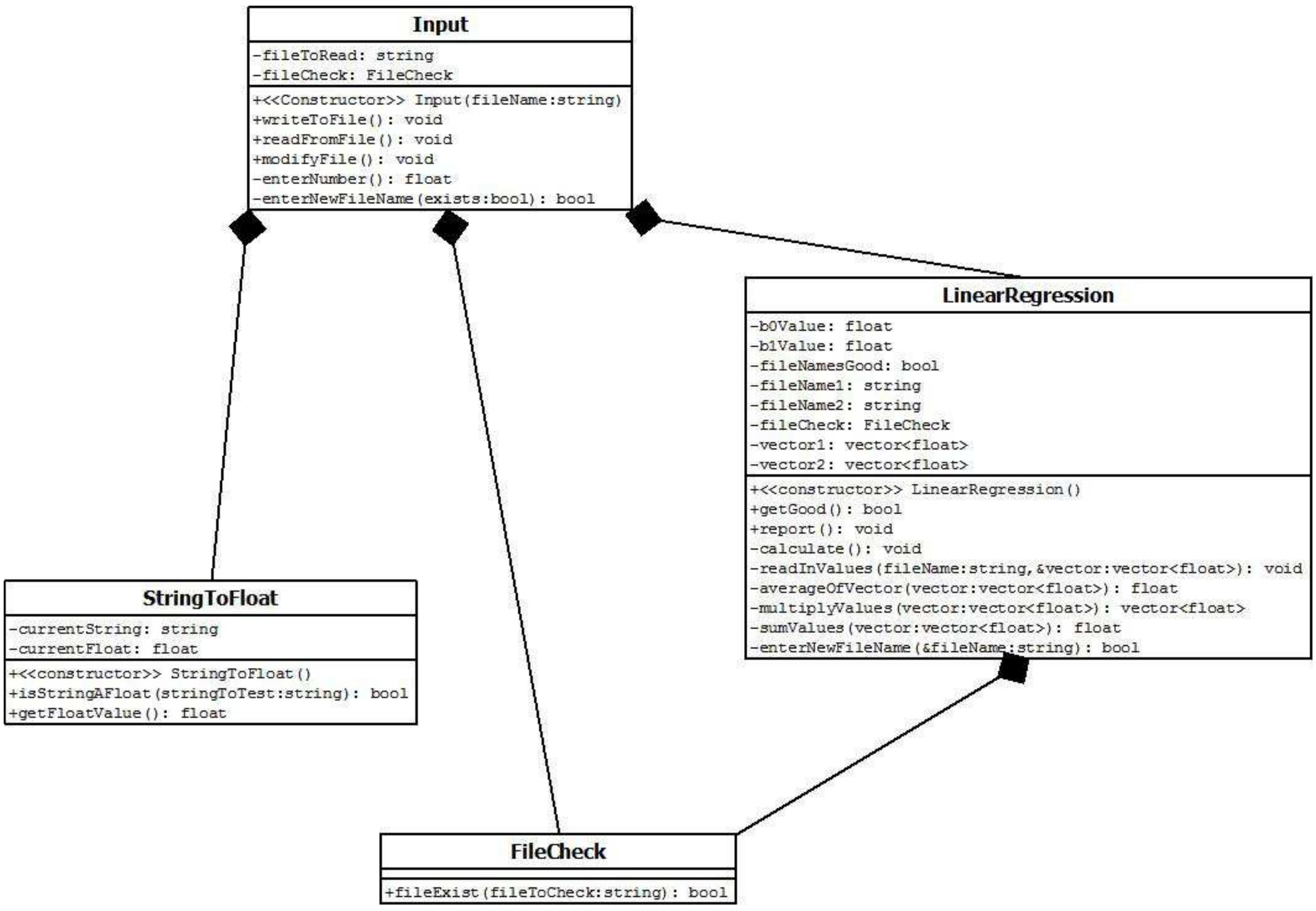
Test 3

```
jamess-imac:program AcousticTime$ ./program4A
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.
Enter 0 to quit.
Choice: 4
Enter the x-axis values filename: xe

Enter the y-axis values filename: yn

B0 = 155.929
B1 = -0.482143
```

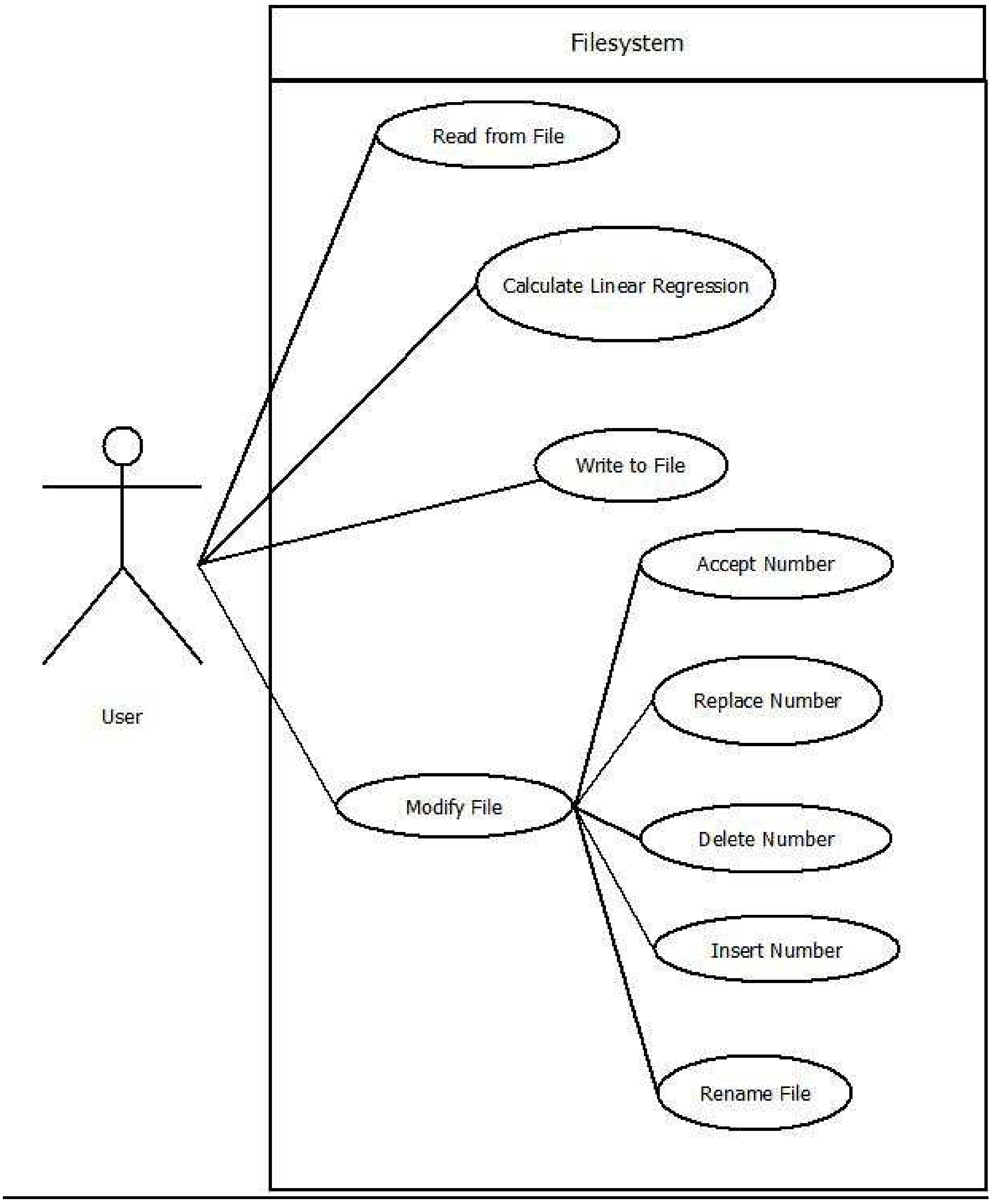
Test 4

```
jamess-imac:program AcousticTime$ ./program4A
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.
Enter 0 to quit.
Choice: 4
Enter the x-axis values filename: xe

Enter the y-axis values filename: yt2

B0 = 101.806
B1 = -0.0229592
```

UML Class Diagram


Input
-fileToRead: string
-fileCheck: FileCheck
+<<Constructor>> Input(fileName:string)
+writeToFile(): void
+readFromFile(): void
+modifyFile(): void
-enterNumber(): float
-enterNewFileName(exists:bool): bool
LinearRegression
-b0Value: float
-b1Value: float
-fileNamesGood: bool
-fileName1: string
-fileName2: string
-fileCheck: FileCheck
-vector1: vector<float>
-vector2: vector<float>
+<<constructor>> LinearRegression()
+getGood(): bool
+report(): void
-calculate(): void
-readInValues(fileName:string,&vector:vector<float>): void
-averageOfVector(vector:vector<float>): float
-multiplyValues(vector:vector<float>): vector<float>
-sumValues(vector:vector<float>): float
-enterNewFileName(&fileName:string): bool
StringToFloat
-currentString: string
-currentFloat: float
+<<constructor>> StringToFloat()
+isStringAFloat(stringToTest:string): bool
+getFloatValue(): float
FileCheck
+fileExist(fileToCheck:string): bool

UML Use Case Diagram



Test	**B0 Expected**	**B1 Expected**	**B0 Actual**	**B1 Actual**
1	-22.55	1.7279	-22.5524	1.72793
2	-23.92	1.4310	-23.9238	1.43097
3	NA	NA	155.929	-0.482143
4	NA	NA	101.806	-0.0229592

```
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
```

```
// 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.\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: 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;
        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);
};
#endif
```

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

#include "LinearRegression.h"
#include <fstream>
#include <iostream>

// Constructor that takes in both file names

LinearRegression::LinearRegression()
{
    fileNamesGood = true;
    b0Value = 0;
    b1Value = 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;
        }

    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);
}

// 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)
{
    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;
}

// 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;
}
```

```
// 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;
}
```