# Assign #6 Source Code:

**#include <vector>**

**#include <string>**

**#include <fstream>**

**#include <iostream>**

**#include <math.h>**

**using** **namespace** std;

**class** **LinReg2** {

**public**:

LinReg2(string filename);

**double** getB0() {**return** B\_0;}

**double** getB1() {**return** B\_1;}

**protected**:

**void** compare ();

**double** B\_0;

**double** B\_1;

**int** N;

**double** AVGX;

vector<**double**> ObjLOC;

vector<**double**> ActLOC;

};

LinReg2::LinReg2(string filename) {

B\_0 = 0, B\_1 = 0;

ifstream in;

in.open(filename.c\_str());

**double** value;

in >> value;

**while** (in.good())

{

ObjLOC.push\_back(value);

in >> value;

ActLOC.push\_back(value);

in >> value;

}

in.close();

}

**void** LinReg2::compare() {

**double** sumX = 0, sumY = 0, sumXY = 0, sumX2 = 0;

**for** (**int** i = 0; i < ObjLOC.size(); i++)

{

sumX += ObjLOC[i];

sumY += ActLOC[i];

sumXY += (ObjLOC[i] \* ActLOC[i]);

sumX2 += (ObjLOC[i] \* ObjLOC[i]);

}

**int** n = ObjLOC.size();

N = n;

**double** sum = (**double**)n;

**double** avgX = sumX / sum;

**double** avgY = sumY / sum;

**double** num = sumXY - sum \* avgX \* avgY;

**double** denom = sumX2 - sum \* avgX \* avgX;

AVGX = avgX;

B\_1 = num / denom;

B\_0 = avgY - B\_1 \* avgX;

}

**class** **RangeFinder** : **public** LinReg2 {

**public**:

RangeFinder(string filename, **double** EstObjLOC):LinReg2(filename)

{

**this**->EstObjLOC = EstObjLOC;

StdDev = 0, T = 0;

compare();

calculateStdDev();

}

**double** calculateRange(**double** TTP);

**double** getStdDev() {**return** StdDev;}

**double** getT() {**return** T;}

**private**:

**void** calculateStdDev();

**int** findT(**double** TTP, **int** DoF);

**double** StdDev;

**double** EstObjLOC;

**double** T;

};

**void** RangeFinder::calculateStdDev() {

**double** sum = 0;

**double** n = (**double**)N;

**for**(**int** i = 0; i < ObjLOC.size(); i++)

sum += pow((ActLOC[i] - B\_0 - B\_1 \* ObjLOC[i]), 2);

StdDev = sqrt((1/(n-2)) \* sum);

}

**Double** RangeFinder::calculateRange (**double** TTP) {

**double** sum = 0;

**double** num = 0;

**double** n = (**double**)N;

**int** code = findT(TTP, N - 2);

**if**( code == 0)

**return** -1;

**for**(**int** i = 0; i < ObjLOC.size(); i++)

sum += pow((ObjLOC[i] - AVGX), 2);

num = pow((EstObjLOC - AVGX), 2);

num = num / sum;

num = num + (1/n) + 1;

num = sqrt(num);

num = num \* T \* StdDev;

**return** num;

}

**int** RangeFinder::findT(**double** TTP, **int** DoF) {

**if**( DoF > 10 || DoF <=0)

{

cout << "number of items is out of range!\n";

**return** 0;

}

**int** col = 0;

ifstream tstream;

tstream.open("tTable.txt");

**double** lowerT, upperT, lowerP, upperP;

tstream.seekg(559);

tstream >> lowerP >> upperP;

**while** (TTP > upperP)

{

lowerP = upperP;

tstream >> upperP;

col++;

}

tstream.seekg((DoF-1)\*7\*8 + col\*7);

tstream >> lowerT >> upperT;

**double** tRange, pRange, tDiff, pDiff;

tRange = upperT - lowerT;

pRange = upperP - lowerP;

pDiff = upperP - TTP;

T = upperT - (pDiff \* tRange / pRange);

tstream.close();

**return** 1;

}

**int** main(**int** argc, **char**\* argv[])

{

string filename;

**if**(argc == 1)

filename = "input.txt";

**else** **if** (argc == 2)

filename = argv[1];

**else**

{

cout << "Invalid Arguement.\n";

**return** 0;

}

**double** expObjLOC = 400;

RangeFinder RF(filename, expObjLOC);

**double** B0 = RF.getB0();

**double** B1 = RF.getB1();

**double** N = B0 + B1 \* expObjLOC;

cout << "y = B0 + B1\*x|| for y is actual new and changed LOC and x is estimated object LOC:\n";

cout << "B0: " << B0 << "\n";

cout << "B1: " << B1 << "\n";

**double** range = RF.calculateRange(**0.85**);

**if**( range > 0)

{

cout << "For and 85 percent prediction range we get:\n";

cout << "StdDev: " << RF.getStdDev() << "\n";

cout << "T: " << RF.getT() << "\n";

cout << "UPI: " << (N + range) << "\n";

cout << "LPI: " << (N - range) << "\n";

}

**else**

cout << "Oh no! failed to find range: invalid p value.\n";

range = RF.calculateRange(**0.95**);

**if**(range > 0)

{

cout << "For a 95 percent prediction range we get:\n";

cout << "StdDev: " << RF.getStdDev() << "\n";

cout << "T: " << RF.getT() << "\n";

cout << "UPI: " << (N + range) << "\n";

cout << "LPI: " << (N - range) << "\n";

}

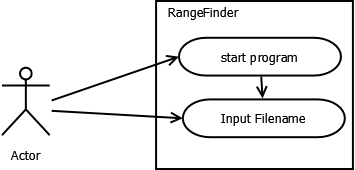
**else**

cout << "Oh no! failed to find range: invalid p value.\n";

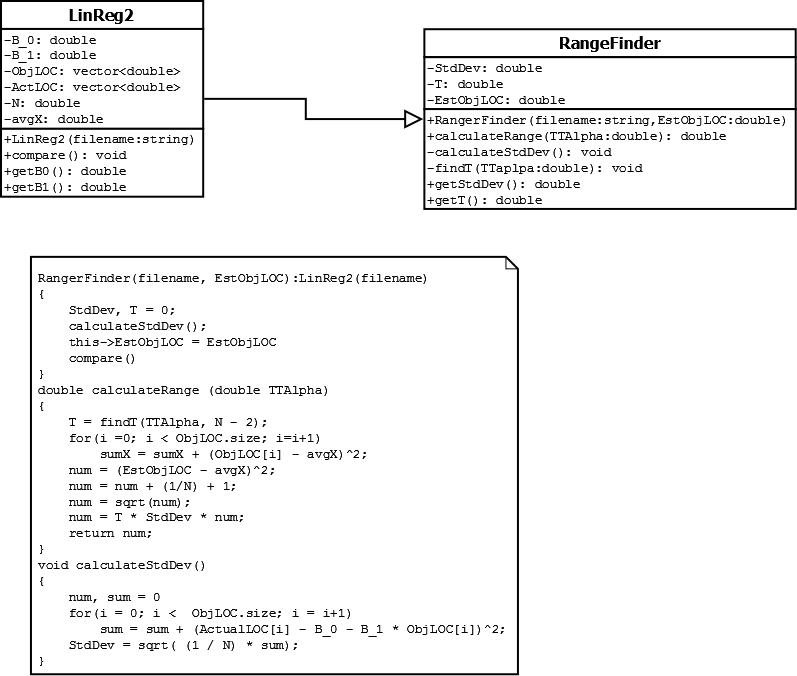
**return** 1;

}

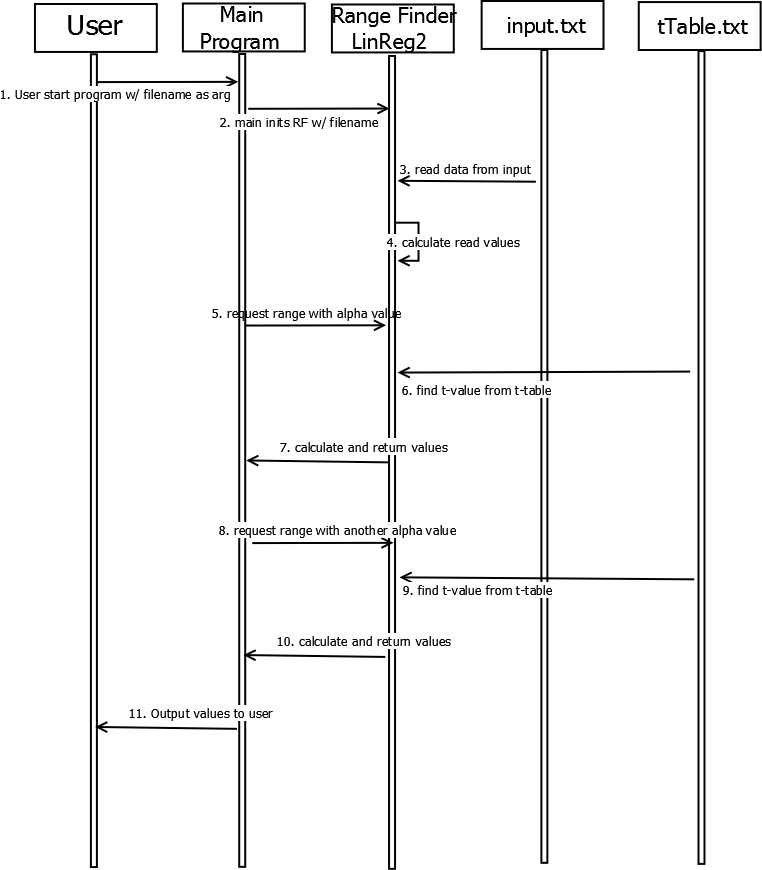
Use-Case Diagram:



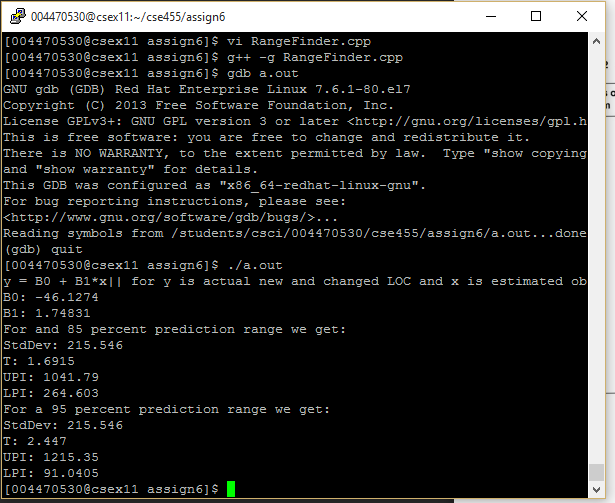
Software Class Diagram:



Sequence Diagram:



Compilation and Execution:



Output Table:

|  |  |  |  |
| --- | --- | --- | --- |
| Test | Parameter | Expected Value | Actual Value |
| Table | Β0 | -46.1252 | -46.1274 |
| Β1 | 1.7483 | 1.74831 |
| UPI(85%) | 1054 | 1042 |
| LPI(85%) | 253 | 265 |
| UPI(95%) | 1216 | 1215 |
| LPI(95%) | 92 | 91 |