CISC2000: Computer Science II LABs Parts 1&2

### This is a multi-part project. Each part builds on the prior part, so they must be completed in order.

### This document describes parts 1 & 2.

#### Introduction:

### These labs calculate inflation. How is this done?

### There is a measure called Consumer Price Index (CPI). This measures how much a ‘basket of goods’ actually costs (on a specific day). Measuring this twice, let’s say at the end of two consecutive months or years, can be used to calculate the inflation rate for that period. Basically, this is the percent change.

Note: Even though money is often stored as double, this lab assumes it is float.

**Simple example:**

Month1CPI = 150. Month2CPI = 156.

Inflation over that month = ((156-150)/150 \* 100) = (6/150) \* 100 = .04 \* 100 = 4%

(Really high!! 48% annual, yikes!)

**Inflation Rate Formula:** ( (newCPI - oldCPI)/oldCPI ) \* 100

#### Inflation Rate Part 1: InflationRate.cpp

This calculates one inflation rate given two CPIs.

### Part 1 example:

**Enter the old and new consumer price indices:** 238.170 239.513

**Inflation rate is 0.563884**

Write a program to read in two consumer price indexes and print out the inflation rate.

Steps:

1. Write a function that takes two floating point values and computes the inflation rate, and returns the result. If either CPI is zero or less, return 0. (Yes, ideally an error should be printed too). For the moment, we’ll assume inflation never lands at precisely zero.

2. In main, read in the two CPI values using the exact text in the above example.

3. Call the function with the two cpis and print the result.

**Additional Test Cases**

* Test for positive inflation rate

**Enter the old and new consumer price indices:** 226.655 227.663  
**Inflation rate is 0.444727**

* Test for negative inflation rate

**Enter the old and new consumer price indices:** 230.085 229.815  
**Inflation rate is -0.11735**

* Test for old CPI negative input

**Enter the old and new consumer price indices:** -230.085 229.815  
**Inflation rate is 0**

* Test for new CPI negative input

**Enter the old and new consumer price indices:** 230.085 -229.815  
**Inflation rate is 0**

* Test for old CPI zero input

**Enter the old and new consumer price indices:** 0 .0001  
**Inflation rate is 0**

### InflationRate Part2: InflationRate2.cpp

Copy your original code to a new version, like this:

cp InflationRate.cpp InflationRate2.cpp

Now we can build on the last program, adding these features:

1. Testing input:

We really want to catch any CPI values that are 0 or less and get them fixed.

Make a new function called getValidCPIValues, which reads in from the user.

If either value is invalid, give an error:

"Error: CPI values must be greater than 0."

and request new values.

1. Multiple Inputs:

Ask the user if they want to make another calculation until they enter ‘n’.

1. Calculate and print the average:

At the end, after they enter ‘n’, calculate and print the average of all the inflation rates. Consider keeping a running total.

**See examples on the next page.**

### Part 2 example:

**Enter the old and new consumer price indices:** 238.654 238.316

**Inflation rate is -0.141633**

**Try again? (y or n):** y

**Enter the old and new consumer price indices:** 0 237.945

**Error: CPI values must be greater than 0**

**Enter the old and new consumer price indices:** 238.316 237.945

**Inflation rate is -0.15567**

**Try again? (y or n):** y

**Enter the old and new consumer price indices:** 237.945 -237.945

**Error: CPI values must be greater than 0**

**Enter the old and new consumer price indices:** 237.945 237.945

**Inflation rate is 0**

**Try again? (y or n):** n

**Average inflation rate is -0.099101**

### Test Case Inputs

* Odd number of test cases

238.343 238.250

y

238.250 237.852

y

237.852 238.031

n

* Even number of test cases

237.805 238.638

y

238.638 238.654

y

238.654 238.316

y

238.316 237.945

n

* Multiple invalid inputs

238.654 238.316

y

0 237.945

238.316 237.945

y

237.945 -237.945

237.945 237.945

n