Name(s) _____

Instructions

1. Collaboration is highly encouraged. Though you may work on this assignment alone, you are encouraged to complete this assignment with a single partner according to the following guidelines:
   - You must work on *all* problems *together*. You must both put in an equivalent effort.
   - You are encouraged to discuss (at a high level) any problems or issues with other individuals or pairs, but you may not directly share code with them.
   - To foster true collaboration, you should use one computer: the person typing is the "driver" while the other person is the "navigator" and should direct the driver. Trade off these roles every 15 minutes or at least every other exercise.
2. Hand in all your source code files through webhandin
   - For convenience, place each program into its own folder with appropriate file name(s)
   - Place all of your files into one folder/directory and zip it up. Turn in the ZIP archive file
3. Be sure to rigorously test your programs with sufficient input examples and test cases.
4. Be sure to thoroughly read this rubric and understand how we will evaluate your program.
5. Printout and hand in a hardcopy of this rubric (first page is sufficient) with you name(s) on it.

Question	Points	Score
1	75	
Total:	75	

1. (75 points) 401k Projection App

In this assignment you will create an application that assists people in saving for a retirement using a tax-deferred 401k program.

Your application will allow a user to enter the following inputs:

- An initial starting balance
- A monthly contribution amount (we'll assume its the same over the life of the savings plan)
- An (average) annual rate of return
- An (average) annual rate of inflation

In addition, you will provide two additional inputs along with two buttons that compute two different scenarios.

- The first will take a number of *years* left until retirement. The app will then compute a monthly savings table which will be a projection out to that many years.
- The second will take a target dollar amount. The app will then compute a monthly savings table until the account balance has reached this target dollar amount.

Take care that the monthly interest rate should be inflation-adjusted. Recall that inflation-adjusted rate of return can be computed with the following formula.

$$\frac{1 + \text{rate of return}}{1 + \text{inflation rate}} - 1$$

To get the monthly rate, simply divide by 12. Each month, interest is applied to the balance at this rate along with the monthly contribution amount.

The details of how your app looks (style) and operates are left for you to design. However, you *must* do at least the following.

- You must define and use at least 3 different functions.
- You must handle errors properly: identify all invalid inputs and display an error message if a user enters them.
- An additional error: as of 2014, annual 401k contributions cannot exceed $17,500. If the user's proposed savings schedule violates this limit, display an error message.

Bonus (25pts)

Use a library to add a graph that depicts the growth in savings. Try graphing the contributions and interest separately. In addition, you should add summaries/totals at the end of the table.

401k Savings Planning

Planning for retirement is extremely important. One of the best ways to save for retirement is to contribute to a 401k Savings Plan. Contributions throughout the year can be made tax free. You can start to withdraw funds at age 59 1/2. Withdrawls are taxed as normal income, but the contributions grow tax-free, increasing your returns.

This app helps you plan for your retirement by asking for some basic inputs and projects your savings and growth. You can target a retirement date or you can target a total amount.

If I start with... $ 10000

And each month I contribute... $ 500

With an annual average return rate of... 9.0 %

And an annual average inflation rate of... 1.2 %

I want to retire in this many years:

10 Compute

I want to retire with this much money:

Compute

Month	Interest Earned	Contribution	New Balance
1	$64.23	$500.00	$10,564.23
2	$67.85	$500.00	$11,132.08
3	$71.50	$500.00	$11,703.58
4	$75.17	$500.00	$12,278.75
5	$78.87	$500.00	$12,857.62
6	$82.58	$500.00	$13,440.20

Figure 1: A screen shot of what the application should look like.

112	$648.40	$500.00	$102,099.40
113	$655.78	$500.00	$103,255.18
114	$663.20	$500.00	$104,418.38
115	$670.67	$500.00	$105,589.05
116	$678.19	$500.00	$106,767.24
117	$685.76	$500.00	$107,953.00
118	$693.37	$500.00	$109,146.37
119	$701.04	$500.00	$110,347.41
120	$708.75	$500.00	$111,556.16

Figure 2: The bottom of the savings table for the example above

Rubric

When we grade your assignment, we will be assessing it based on the items below.

Following Instructions

- All required soft-copy files handed in via webhandin
- Correct file name(s) and organization
- Any required hardcopies (this rubric) handed in
- Programs successfully execute and there are no formatting or syntax errors

Style

- Appropriate variable and function/method identifiers
- Style and naming conventions are consistent
- Good use of whitespace; proper indentation
- Clean, readable code
- Code is well-organized

Documentation

- Well written comments that clearly explain the purpose of each non-trivial piece of code
- Comments explain the "what" and "why"
- Comments are not overly verbose or overly terse
- Code itself is "self-documenting"; explains the "how"

Program Design

- Code is well-organized and efficient
- Code is modular; substantial pieces of it could be reused; few redundancies
- Code is easily understood and maintainable
- It is clear that sufficient testing has been performed
- Corner cases and bad input have been anticipated and handled appropriately

Program Correctness

- Source code compiles, executes as expected
- Program runs as specified: correctly reads any input; correctly formatted output
- Test cases successfully execute