Names _____

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	100	
Total:	100	

1. (100 points) Cost of Food App

The United States Department of Agriculture (USDA) regularly publishes estimates for the cost of food for US families. This data is updated monthly and is available at `http://www.cnpp.usda.gov/USDAFoodPlansCostofFood`. However, the estimates are not presented in an easy to calculate manner. Estimates are given for individuals based on age and gender, but it would require a non-trivial amount of work to calculate an estimate for a given family.

You will make the process of estimating food cost and presenting this data easier by creating an app that will allow users to enter data for their family and produce an accurate estimate based on the USDA's data. In particular you will create a webform that will allow a user to enter details on the composition of their family (number and age of children, number of male/female adults and their age) and which type of plan (thrifty, low-cost, moderate, liberal) they want to estimate. The user will then be presented a *monthly* meal plan cost based on this input. Be sure to take into account the third footnote where an adjustment must be made for families larger or smaller than 4 individuals.

We want you to have a lot of freedom with how you go about doing this project. However, there are a few minimal things you must do:

- Your form should be dynamic: a user should be able to indicate the number of children and adults and then be presented with exactly that many input boxes for the gender, age, etc.
- In addition to the final amount, you should provide *some* visualization of the result (a dynamic pie chart of the cost breakdown, a bar graph comparison of plan levels, etc.); the choice of visualization is up to you.
- In addition, provide a table of the projected annual cost of food over the next 10 years assuming an inflation rate (for food) of .8% (or you may use the most recent Consumer Price Index, CPI for food available at `http://www.bls.gov/cpi/`).
- You must make the application look visually appealing.

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