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	25	
2	25	
3	25	
Total:	75	

Programs

1. (25 points) Gas with ethanol is generally cheaper, but might be less efficient. This means that the actual per-mile cost of gasoline may differ between ethanol and regular. To determine which one is the better deal, write a program that prompts the user for the cost of regular gas (dollars per gallon) as well as its efficiency in the user's car (miles per gallon). Then prompt for the cost and efficiency of the ethanol gas. Compute the actual cost for each types (dollars per mile) and report both figures to the user. Include a message indicating which type of gas (regular or ethanol) is the better deal.

   For example, if the user inputs a cost/efficiency for regular as \$2.70 per gallon and 25 miles per gallon and \$2.20 per gallon and 20 miles per gallon respectively, the program should report the result similar to the following.

   ```
   Regular Cost: $0.108 per mile
   Ethanol Cost: $0.11 per mile
   Regular gasoline is the better deal.
   ```

2. (25 points) When taking out a loan for a given *principle*, a customer pays a certain percentage of interest (APR) over a certain *term* (number of months). These determine a fixed payment that the customer will pay on a monthly basis. In each monthly payment, a certain amount is applied to the accrued interest and the rest toward the principle.

   The monthly payment amount can be determined by the following formula:

   $$m = p\left(\frac{r}{1-(1+r)^{-t}}\right)$$

   where

   - m is the resulting monthly payment
   - p is the original principle amount borrowed
   - t is the term (in number of months)
   - r is the monthly interest rate which is the APR divided by 1200 (the APR is on the scale $[0, 100]$ so we divide it by 100 then divide by 12 to get the monthly rate)

   In addition, we can compute the *true* cost of a loan which is the amount that the customer pays in interest over the life of the loan. This is simply the total amount of all payments less the original principle.

   Write a JavaScript program that prompts the user for the necessary inputs. It should then compute the monthly payment amount and the true cost of the loan. Output a summary of the loan to the console or in an alert box. Take care: if the user enters invalid data, instead of a summary, you should output an appropriate error message instead.

   For example, if a user inputs $p = \$10,000$, $t = 60$ (5 years), and $r = 4.9\%$, the output could look *something* like the following.

```
Loan Amount:     $10000
Months:           60
APR:              4.9%
Monthly Payment: $188.25
Total Cost:      $11295.00
True Cost:       $1295.00
```

3. (25 points) Write an app to help people track their cell phone usage. Cell phone plans for this particular company give you a certain number of minutes every 30 days which must be used or they are lost (no rollover). We want to track the average number of minutes used per day and inform the user if they are using too many minutes or can afford to use more.

   Write a program that prompts the user to enter the following pieces of data:

   - Number of minutes in the plan per 30 day period, m
   - The current day in the 30 day period, d ($1 \leq d \leq 30$)
   - The total number of minutes used so far u

   The program should then compute whether the user is over, under, or right on the average daily usage under the plan. It should also inform them of how many minutes are left and how many, on average, they can use per day for the rest of the month. Of course, if they've run out of minutes, it should inform them of that too.

   For example, if the user enters $m = 250$, $d = 10$, and $u = 150$, your program should print out something similar to the following.

```
10 days used, 20 days remaining
Average daily use: 15 min/day

You are EXCEEDING your average daily use (8.33 min/day),
continuing this high usage, you'll exceed your minute plan by
200 minutes.

To stay below your minute plan, use no more than 5 min/day.
```

   Of course, if the user is under their average daily use, a different message should be presented. You are allowed/encouraged to compute any other stats for the user that you feel would be useful.

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