Assignment SPIM 2

Due Date: April 9

Purpose

Finally, more SPIM! More fun than the start of baseball season! In this problem, you will manipulate floats and characters, and write a couple of functions (procedures). You may have done a very similar problem in COMP 115, so how hard could it be?

Problem

Cowberland Farms has hired you to test out their new gas pumps. You need to simulate the workings of the payment system that will be installed on the pumps. This system accepts payment either by traditional credit card or through the SmahtPay debit app. The pumps use a simple processor that must be programmed in assembly language, because there is no compiler available with that hardware for any high-level language.

Input

You may assume that all input will be correct. We are simulating a gas pump that has hard-coded buttons, so the customer can not type in an incorrect value.
The program should first prompt for the payment type, which will be a character. An input of 'd' signifies SmahtPay ('d' for "debit"); a 'c' indicates a credit purchase; a 'q' means the pump simulation is to end.
The second prompt is for the gas type, also a character. This character represents the gas type and price as shown below:

Type	Code	Price
Regular	r	2.619
Plus	p	2.819
Super	s	2.959

Finally, the program should prompt for the number of gallons, a float.

After the result is displayed (described below), the program should repeat.

Output

The output should be a "receipt" that shows all of the input information and the final cost of the gas that is billed to the credit card or to the SmahtPay account. If the customer uses SmahtPay, the gas cost should be reduced by 10 cents/gallon. In all cases, **the total should be rounded to the nearest penny.** You may want to check out how real pumps show this information. All output should be in a clearly readable format, aligned as best you can. For example, if 8.0 gallons of Super were purchased with a credit card, the output might look like:

```
Fuel type:    Super
Gallons:      8.0
Cost/gallon:  2.959
Credit/debit: Credit
Total due:    23.67
```

When the program ends, a message to the effect that the pump is shut down should be displayed.

Specifics

- Your program should include at least two functions (procedures): One should prompt for the gas type and return the price per gallon. The second should be passed the number of gallons and the cost per gallon and display the total cost rounded to the nearest penny. Feel free to add additional functions/procedures.
- Rounding to the nearest penny is non-trivial. One way to do this is to use the instruction to convert to integer to truncate extra decimal places. **Note that floats converted to integers must still be stored in floating point registers.** This is annoying, but you can store an integer that is currently in a floating point register in memory as a word, using the `s.s` instruction; alternatively, you can use the `mfc1` instruction to move a value into an integer (word) register. Storing the value in one of these ways will ensure that extra decimal places are indeed truncated. Then the trick is to display the final cost as integers rather than a floating point value so that you will see only two decimal places. SPIM/MIPS has no formatting capabilities.
- All dollar values should be displayed with only **two significant decimal places**.[1] No extra zeros (or other digits) should be displayed.
- The program should loop until the sentinel ('q') is input.
- Follow the style conventions as in the first assignment. Comment all functions (procedures); give a one-line description and identify the parameter(s) and return value(s).

Notes

- Once again, the general problem is trivial. But the rounding to two decimal places is not straightforward in SPIM, and will take some time to do properly. Write the program in Python/C++ **without** using those languages' formatting functionality, and then convert to SPIM.
- The character input is also a bit annoying, but we've done something similar in lab.
- Remember how long things take to program in SPIM; don't wait too long to start!
- Add the Honor Code in a comment near the top that includes your electronic "signature." Turn in your source code via email by 11:59:59 PM on the due date. Use the naming convention as follows: your first initial followed by your last name and finally "SPIM2.s", as in `mgousieSPIM2.s`. Note there are **no spaces** in the filename.

The computer is no better than its program.
– Elting Elmore Morison, in *Men, Machines and Modern Times* [1966]

And this `xspim` *program is pretty bad.*
– Anonymous

[1] If the total comes out to a whole number, it is ok to display just one zero after the decimal place.