Exercise UVa #10137 (The Trip). The problem is described at:

http://uva.onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&page=show_problem&problem=1078.

Lessons Learned

- I spent more time thinking than I did actual programming — and this had a nice payoff: my program was accepted by the judge on the second submission.
- **Floating point arithmetic requires great care.** Because every penny matters in this problem, it is much safer to work with integer types.
- A few students asked me for help in finding a counter-example; i.e., could we find a data set that would show incorrect output for their program. This turned out to be much more difficult than I expected. **Devising good test cases is hard work!**
- Programming—in C or any other language—is an exacting and challenging activity. Good planning and sound algorithms are required. The fact that our programs must correctly respond to an infinite possibility of inputs is one of the most challenging aspects.

Explanation

As a starting point, let's consider a few concrete examples. Suppose five students go on a trip with the following costs, in cents:

$$20 \quad 18 \quad 16 \quad 4 \quad 2$$

This is the easiest situation, since the average cost is $(20+18+16+4+2)/5 = 60/5 = 12$. (This is the easiest situation because 60 is evenly divisible by 5 — there are no remaining pennies to worry about.) To find out how much money must change hands, look at it from the perspective of all the students who overpaid. In this example, three people overpaid and require reimbursements of $20-12$, $18-12$ and $16-12$, for a total of $8+6+4 = 18$.

For a second example, consider these costs:

$$20 \quad 18 \quad 16 \quad 5 \quad 3$$

The total cost is now 62, giving an average of $62/5$ — a quotient of 12 with remainder 2. To equalize costs, three students will pay 12 and two students will pay 13. To minimize the amount of money which changes hands, we will penalize two of students who are already over the average, asking them to pay an extra penny.

Here's a slightly different way to think of it. Let everyone pay $\lfloor 62/5 \rfloor$, then distribute the extra cost (to the over-payers) from a pool of $62 \bmod 5 = 2$ pennies. As in the first example, we can account for the money that needs to trade hands from the perspective of those who overpaid. As a first step, ignore the pool of extra costs. The three people who overpaid would expect reimbursements of $20-12$, $18-12$, and $16-12$. We adjust this by considering the pool of 2 pennies, asking two of the three to pay one penny more. The total reimbursement is thus $8+6+5-1-1 = 16$. (We will decrease the pool by one penny each time we consider an over-payer; when the pool reaches 0, any remaining over-payers will not be assessed the extra penny cost.)

As a third example, suppose we have these costs:

$$62 \quad 0 \quad 0 \quad 0 \quad 0$$

In this case, as before, we have an average of 12 with a pool of 2 cents. The amount to redistribute is thus $(62-12)-1 = 49$. This example shows that the pool is not necessarily reduced to 0 as a result of considering all the over-payers.

Coding Details

As already mentioned, performing all arithmetic with integer values is the easiest way of maintaining accuracy to the penny. Although the input values "look like" they should be read as floating point values, we can consider them to be pairs of integers and use C's `scanf` format specifier to read a pair of integers from each line of the input. In the following snippet of code, all the variables are of type `int`.

```
/* Collect and save the data for the current trip */
for (i = 0; i < n; i++) {
  scanf("%d.%d", &d, &c);
  amt[i] = 100*d + c;
```

As an alternative, we could read the inputs as a floating point type, representing dollars and cents, and then convert to an integer type representing the number of cents. (For example, 1.23 would be converted to 123.) Due to possible errors involved in storing floating point values, we need to carefully round when the conversion to an integer type is performed:

```
double amtPaid;          /* amount one student paid */

/* Collect and save the data for the current trip */
for (i = 0; i < n; i++) {
  scanf("%lf", &amtPaid);
  amt[i] = 100*amtPaid + 0.5; /* number of cents paid by this student */
```

The complete program is shown on the following page.

Source Code

```c
/*
  Name:     trip.c
  Purpose: Solution to Programming Challenges "The Trip" (# 10137)
  Author:   Bill Slough
  Date:     January 21, 2012
*/

#include <stdio.h>

#define MAX_STUDENTS 1000

int main() {
    int d, c;               /* dollars and cents */
    int n;                  /* number of students for any one trip */
    int i;                  /* loop index */

    int amt[MAX_STUDENTS];  /* amount each student paid, in cents */
    int total;              /* total amount, in cents, for one trip */

    int average;            /* amount each student should pay for trip,
                               not counting a possible extra cent */
    int leftover;           /* number of cents to distribute among students
                               to make the total exact */

    int toDistribute;       /* amount of money to distribute to share the costs */

    scanf("%d", &n);
    while (n != 0) {
        /* Collect and save the data for the current trip */
        for (i = 0; i < n; i++) {
            scanf("%d.%d", &d, &c);
            amt[i] = 100*d + c;
        }

        /* Determine the total amount paid for this trip */
        total = 0;
        for (i = 0; i < n; i++) {
            total = total + amt[i];
        }

        /* Determine the average amount */
        average = total/n;      /* the FLOOR of the true average */
        leftover = total % n;

        /* Examine what each student paid; determine the total amount to be repaid. */
        toDistribute = 0;
        for (i = 0; i < n; i++) {
            if (amt[i] > average) {
                /* this student overpaid and needs reimbursement */
                toDistribute = toDistribute + (amt[i] - average);

                /* To keep reimbursements lower, increase the cost
                   of the overpayers by one penny */
                if (leftover > 0) {
                    toDistribute = toDistribute - 1;
                    leftover = leftover - 1;
                }
            }
        }

        /* Announce the minimum amount which needs to be redistributed */
        printf("$%d.%02d\n", toDistribute/100, toDistribute % 100);

        /* Get ready for the next group... */
        scanf("%d", &n);
    }

    return 0;
}
```