WORKED EXAMPLE 5.3

Calculating a Course Grade

© paul kline/iStockphoto.

Students in this course take four exams and earn a letter grade (A+, A, A–, B+, B, B–, C+, C, C–, D+, D, D–, or F) for each of them. The course grade is determined by dropping the lowest grade and averaging the three remaining grades. To average grades, first convert them to number grades, using the usual scheme A+ = 4.3, A = 4.0, A– = 3.7, B+ = 3.3, ..., D– = 0.7, F = 0. Then compute their average and convert it back to the closest letter grade. For example, an average of 3.51 would be an A–.

Problem Statement Your task is to read inputs of the form:

letter_grade1 letter_grade2 letter_grade3 letter_grade4

For example,

```
A- B+ C A
```

For each input line, your output should be

letter_grade

where the letter grade is the grade earned in the course, as just described. For example,

```
A-
```

The end of inputs will be indicated by a *letter_grade1* value of `Q`.

Step 1 Carry out stepwise refinement.

We will use the process of stepwise refinement. To process the inputs, we can process each line individually. Therefore, we define a task *process line*.

To process a line, we read the first grade and bail out if it is a `Q`. Otherwise, we read the four grades. Since we need them in their numeric form, we identify a task *convert letter grade to number*.

We then have four numbers and need to find the smallest one. That is another task, *find smallest of four numbers*. To average the remaining ones, we compute the sum of all values, subtract the smallest, and divide by three. Let's say that is not worth making into a subtask.

Next, we need to convert the result back into a letter grade. That is yet another subtask *convert number grade to letter*. Finally, we print the letter grade. That is again so simple that it requires no subtask.

Step 2 Convert letter grade to number.

How do we convert a letter grade to a number? Follow this algorithm:

```
grade_to_number(grade)
ch = first character of grade
If ch is A / B / C / D / F
   Set result to 4 / 3 / 2 / 1 / 0.
If the second character of grade is +
   Add 0.3 to result.
If the second character of grade is -
   Subtract 0.3 from result.
Return result.
```

Here is a function for that task.

```
/**
   Converts a letter grade to a number.
   @param grade a letter grade (A+, A, A–, ..., D–, F)
   @return the equivalent number grade
*/
double grade_to_number(string grade)
{
   double result = 0;
   string first = grade.substr(0, 1);
   if (first == "A") { result = 4; }
   else if (first == "B") { result = 3; }
   else if (first == "C") { result = 2; }
   else if (first == "D") { result = 1; }
   if (grade.length() > 1)
   {
      if (grade.substr(1, 1) == "+")
      {
         result = result + 0.3;
      }
      else
      {
         result = result - 0.3;
      }
   }
   return result;
}
```

Step 3 Convert number grade to letter.

How do we do the opposite conversion? Here, the challenge is that we need to convert to the *nearest* letter grade. For example, suppose x is 2.9. This value is closer to a B (23.0) than to a B– (2.7). When a value is exactly in the middle between two letter grade values, we give the higher grade. For example, 2.85 is rounded up to a B, and 3.15 is rounded up to a B+ (3.3). That is, any grade that is at least 2.85 but less than 3.15 is a B.

We can make a function with 13 branches, one for each valid letter grade.

```
/**
   Converts a number to the nearest letter grade.
   @param x a number between 0 and 4.3
   @return the nearest letter grade
*/
string number_to_grade(double x)
{
   if (x >= 4.15) { return "A+"; }
   if (x >= 3.85) { return "A"; }
   if (x >= 3.5) { return "A-"; }
   if (x >= 3.15) { return "B+"; }
   if (x >= 2.85) { return "B"; }
   if (x >= 2.5) { return "B-"; }
   if (x >= 2.15) { return "C+"; }
   if (x >= 1.85) { return "C"; }
   if (x >= 1.5) { return "C-"; }
   if (x >= 1.15) { return "D+"; }
   if (x >= 0.85) { return "D"; }
   if (x >= 0.5) { return "D-"; }
   return "F";
}
```

Exercise P5.5 suggests an alternate approach.

Step 4 Find the minimum of four numbers.

Finally, how do we find the smallest of four numbers? Let's suppose we can find the smallest of two numbers, with a function `min(x, y)`. Then the smallest of four numbers is `min(min(x1, x2), min(x3, x4))`. Finding the smallest of two numbers is easy:

```
/**
   Returns the smaller of two numbers.
   @param x a number
   @param y a number
   @return the smaller of x and y
*/
double min(double x, double y)
{
   if (x < y)
   {
      return x;
   }
   else
   {
      return y;
   }
}
```

Step 5 Process a line.

As previously described, to process a line, we read in the four input strings, convert grades to numbers, and compute the average after dropping the lowest grade. Then we print the grade corresponding to that average.

However, if we read the first input string and find a `Q`, we need to signal to the caller that we have reached the end of the input set and that no further calls should be made.

Our function will return a `bool` value: `true` if it was successful, `false` if it encountered the sentinel.

```
/**
   Processes one line of input.
   @return true if the sentinel was not encountered
*/
bool process_line()
{
   cout << "Enter four grades or Q to quit: ";
   string g1;
   cin >> g1;
   if (g1 == "Q") { return false; }
   string g2;
   string g3;
   string g4;
   cin >> g2 >> g3 >> g4;
   double x1 = grade_to_number(g1);
   double x2 = grade_to_number(g2);
   double x3 = grade_to_number(g3);
   double x4 = grade_to_number(g4);
   double xlow = min(min(x1, x2), min(x3, x4));
   double avg = (x1 + x2 + x3 + x4 - xlow) / 3;
   cout << number_to_grade(avg) << endl;
   return true;
}
```

Step 6 Write the `main` function.

The `main` function is now utterly trivial. We keep calling `process_line` while it returns `true`.

```
int main()
{
   while (process_line())
   {
   }
   return 0;
}
```

EXAMPLE CODE Here is the complete program.

worked_example_3/grades.cpp

```
#include <iostream>
#include <string>

using namespace std;

/**
   Converts a letter grade to a number.
   @param grade a letter grade (A+, A, A-, ..., D-, F)
   @return the equivalent number grade
*/
double grade_to_number(string grade)
{
   double result = 0;
   string first = grade.substr(0, 1);
   if (first == "A") { result = 4; }
   else if (first == "B") { result = 3; }
   else if (first == "C") { result = 2; }
   else if (first == "D") { result = 1; }
   if (grade.length() > 1)
   {
      if (grade.substr(1, 1) == "+")
      {
         result = result + 0.3;
      }
      else
      {
         result = result - 0.3;
      }
   }
   return result;
}

/**
   Converts a number to the nearest letter grade.
   @param x a number between 0 and 4.3
   @return the nearest letter grade
*/
string number_to_grade(double x)
{
   if (x >= 4.15) { return "A+"; }
   if (x >= 3.85) { return "A"; }
   if (x >= 3.5) { return "A-"; }
   if (x >= 3.15) { return "B+"; }
   if (x >= 2.85) { return "B"; }
   if (x >= 2.5) { return "B-"; }
   if (x >= 2.15) { return "C+"; }
   if (x >= 1.85) { return "C"; }
   if (x >= 1.5) { return "C-"; }
```

```
49    if (x >= 1.15) { return "D+"; }
50    if (x >= 0.85) { return "D"; }
51    if (x >= 0.5) { return "D-"; }
52    return "F";
53 }
54
55 /**
56    Returns the smaller of two numbers.
57    @param x a number
58    @param y  a number
59    @return the smaller of x and y
60 */
61 double min(double x, double y)
62 {
63    if (x < y)
64    {
65       return x;
66    }
67    else
68    {
69       return y;
70    }
71 }
72
73 /**
74    Processes one line of input.
75    @return true if the sentinel was not encountered
76 */
77 bool process_line()
78 {
79    cout << "Enter four grades or Q to quit: ";
80    string g1;
81    cin >> g1;
82    if (g1 == "Q") { return false; }
83    string g2;
84    string g3;
85    string g4;
86    cin >> g2 >> g3 >> g4;
87    double x1 = grade_to_number(g1);
88    double x2 = grade_to_number(g2);
89    double x3 = grade_to_number(g3);
90    double x4 = grade_to_number(g4);
91    double xlow = min(min(x1, x2), min(x3, x4));
92    double avg = (x1 + x2 + x3 + x4 - xlow) / 3;
93    cout << number_to_grade(avg) << endl;
94    return true;
95 }
96
97 int main()
98 {
99    while (process_line())
100   {
101   }
102   return 0;
103 }
```