]>

Assignment: Triangle

Write a program which reads in three integers representing the lengths of the sides of a triangle, and then prints out information about the triangle. You are given this skeleton as a starting point:

> triangle.c

The program takes its input from the command line, and prints its output on the standard output, e.g.

```
./triangle 3 4 5
Equilateral
```

The possible responses are:

```
Equilateral     (all sides equal)
Isosceles       (two sides equal)
Right           (has a right angle)
Scalene         (sides unequal, but no right angle)
Flat            (no area)
Impossible      (lengths don't add up)
Illegal         (lengths are invalid)
```

The program includes 50 auto-tests, of which 49 are currently commented out. If the program is run with no command line arguments, it runs the tests and prints the number of tests passed. That number is your mark for the first part of the assignment.

Your aim is to make the program work with all the tests uncommented. You should change the content of the `triangle` function, and add further functions above it as you think fit. You shouldn't change the overall design, because we will test your `triangle` function directly from the outside for marking purposes.

Notes

The first thing to do is to compile and run the program - the first test should pass. Then use cut-and-paste in your editor to move the second test out of the `/*...*/` comment. Compile and run the program again. Keep doing this for all the tests. If a test doesn't pass, develop the program further until it does.

A suggested design is for the triangle function to convert the strings to integers, and swap the integers around until the first is the largest. Then one extra function can be written to check each of the properties the triangle might have, and then those functions can be called in a suitable order.

Standard library functions from the three modules `stdio`, `stdlib` and `string` can be used. The ones you are most likely to need are:

- `atoi(s)` converts a string `s` to an int
- `sprintf(s,"%d",n)` converts an int `n` to a string, provided you have allocated space for the string with, say, `char s[100]`.
- `strcmp(s1,s2)` compares two strings alphabetically, returning zero if they are equal, a negative number if `s1` is less than `s2`, and a positive number otherwise

Testing whether a length is valid or not can be done in various different ways. One way involves writing a loop. Another uses the three library functions above.

The last few tests are designed to catch overflow problems. This is when integers exceed the limit of the `int` type. You might consider switching to the `long` type in parts of the program where you think overflow might occur (if you copy an `int` variable into a `long` variable, the right thing happens). It is OK for this assignment to assume that `int`s have 32 bits and `long`s have 64 bits.

Extension

If you have time, and if you are interested or you want more than 50% for the assignment, you can try this extension. Write any program you want which involves simple classification or calculation.

As an example, you might write a program `grade.c` which takes a mark for a unit and outputs a grade band, with its grade description, according to the aside on marks.

You can add features to this as you feel fit, but staying within university conventions. For example, the unit mark can have one decimal place, or there can be many unit marks to be averaged, or each unit mark can be accompanied by its number of credits for a weighted average to be calculated, or grades could be converted to or from the Grade Point Average system.

Submit the source file, and a file `readme.txt` (or a pdf file) which describes your aims for the program and how far you got with it. There are no marks for the report as such, but you may not get any marks for the program without the report, because we need it for marking.

A mark out of 50 for the extension will be awarded by swiftly reading the report, checking whether your program matches what you claim, judging the sophistication of what has been done, and checking whether the program follows the conventions and advice in the lectures. The mark will aim to make your total for the assignment meet the university scale. So assuming you get full marks for the triangle program, 10/50 means "this raises your total result from average to good", 20/50 means "overall first class", 30/50 means "overall above and beyond what was expected" and 40/50 means "publishable in a research journal".