CMSC 106 Project #3 Fall 2010

Due date: Tuesday, October 19, 2010

1 Purpose

In this project you will write a program using loops and nested loops. Although as a matter of style and clarity it is usually extremely important to determine the most appropriate type of loop for each iterative task in a program, in order to get practice with all of C's loop statements it is suggested that you intentionally use some of each type in your project.

You are on the committee for this Winter's Olympic games in Vancouver. You are hosting a fund raiser and want to display flags of several countries to encourage people to give money toward your cause. Your program is to create images of flags for these for decorations. (Your program at this point is only going to do 4 different flags, but those flags are available in a variety of sizes.) Not having access to a fancy color graphics terminal, you've decided to start by writing a program which produces facsimiles of flags using letters to indicate which colors they contain.

2 Project description

As you read this section, you may also want to look at the `infile` and `outfile` as a sample of what the `main.c` enclosed. The output should look like `outfile` when the main provided is run using the lines of `infile`. The output of your program should match the `outfile` exactly.

The input to your program is to consist of a sequence of input lines with no prompt requesting them. Each of the lines will have two integers, the first representing a country code and the second a flag height. Your program will read each input line and then print a flag of the chosen country which is of the size that was entered. Your program is to stop reading input lines and quit once an input line is read which has a country code of 0. That last line will be complete (contain 2 values), but the country code of 0 is what should terminate the program.

Valid country codes, and the countries they represent, are 1 for Mauritius, 2 for Jersey (without the crest), 3 for Eritrea (without its crest), and 4 for Afghanistan (without the crest). The flags for these nations appear on the class web site in color, but their descriptions (with details of how you will need to display them in their horizontal orientation) follow:

Mauritius (in Africa)	4 horizontal parallel stripes of color All equal in size Parallel to the long direction of the flag From top to bottom: Red, Blue, Yellow and Green
Jersey (in Europe)	A mostly white flag with a red X going from corner to opposite corner. The red X should be three characters wide regardless of the size of the flag. Note: completely skip the crest that appears on the color version of their flag.
Eritrea (in Africa)	A rectangle which is broken into three triangles One triangle is red in color, is an isosoles triangle and has its base taking the whole left side of the flag reaching all of the way to the middle of the other side. The second is a green triangle which covers the top right corner (almost reaching the top left corner to the middle of the right). The third is a blue triangle which covers the bottom right corner (almost reaching the bottom left corner to the middle of the right). Note: the far left of the flag should be three wide regardless of the size of the flag. Note: completely skip the gold crest which also appears on their flag.
Afghanistan	Three vertical parallel stripes of color All equal in size Parallel to the short direction of the flag From left to right: black, red and green Note: again - skip the crest

Examples:
For Mauritius size 4:

```
------------
BBBBBBBBBBBB
YYYYYYYYYYYY
GGGGGGGGGGGG
```

For Jersey size 5:

```
---WWWWWWWWW---
WWW---WWW---WWW
WWWWWW---WWWWWW
WWW---WWW---WWW
---WWWWWWWWW---
```

For Eritrea size 7:

```
---GGGGGGGGGGGGGGGGG
---------GGGGGGGGGGG
---------------GGGGGG
--------------------
---------------BBBBBB
---------BBBBBBBBBBB
---BBBBBBBBBBBBBBBBB
```

For Afghanistan size 5:

```
BBBBB-----GGGGG
BBBBB-----GGGGG
BBBBB-----GGGGG
BBBBB-----GGGGG
BBBBB-----GGGGG
```

All the colors on the flags are to be represented using their uppercase first letter (e.g. 'B' for blue or Black, 'W' for white, etc.) with the one exception that the Red color of the flag should be done in a "-" (dash or minus sign) which will make it more obvious. The flag will be wider on the screen than it is high – always exactly 3 times the height. The height is always the value given by the user.

If any input line contains an incorrect country code your program must print an explanatory error message and read a new input line without printing a flag for that line. This explanatory error message must contain the exact word "ERROR" followed on the same line by the invalid input value which is then followed by the two word sequence "Country Code". This is done by the function that is reading in the country code.

In order to display a flag, the specified size must satisfy certain restrictions. All flag sizes must be less than or equal to 20 and greater than or equal to 3. In addition, certain size restrictions apply to each country's flag. The Maruitius flag size must be a multiple of 4. The Jersey and Eritrea flag sizes must be odd. Afghanistan has no additional requirement. When an invalid size is given for a flag, your program must print an explanatory error message instead of the flag. This error message must begin with the exact word "ERROR", followed by the invalid input value, and then followed by the word "Size". E.g., if 9 was given as the size for Mauritius, then (since 9 is not a multiple of 4) the program would output "ERROR 9 Size". When writing the method for that one flag, you can assume the height passed is appropriate for that flag since the function that gets the height from the user will give an error message if the height given is not appropriate.

Before each set of output (flag or error message), a line must be printed to tell the person who is reading the output which input line the following output corresponds to. The line must first have an integer (starting with 1 for the first input line, 2 for the second input line, etc.) followed immediately by a series of exactly 10 stars (asterisks). This is done by the `main.c` given.

3 Project requirements

All your C programs in this course should be written in ANSI C, which means they must compile and run correctly with `gcc -Wall -Werror -ansi -pedantic-errors` on the grace system as was setup at the beginning of the course.

You will lose credit if your program generates any warning messages when it is compiled. Even if you already know what they are, you may not use any C language features other than those introduced so far in lecture or lab. Note that as a arrays and structs may not be used. In addition, neither the `goto` nor the `continue` statement may be used, and the `break` statement may not be used in any loop. Using C features not presented in the lectures before the date of the assignment, or using the `goto` statement, or `break` or `continue` in loops **will result** in losing credit.

You may not modify the `main.c` or the `flags.c` file at all. All of your code must appear in the `flags.c` file. You may add other methods or constants to the `flags.c` if you would like, but they will not have a prototype in the `flags.h`.

All work for this project must be your own individual work. You may not work with any other student or compare code in any way. If you need help see either the instructor or TA. Obtaining help from others will be considered cheating. Getting code from a web page or other location will be considered plagerism.

Your program must have a comment near the top which contains your name, login ID, student ID, your section number, and an original description of the action and operation of the program. Your program should be written using good programming style and formatting, as discussed in class and throughout your textbook. For this project, style is considered to consist of:

- You must implement each of the functions defined in the flags.h file without changing the name, return type, parameter(s) or what it does.
- adequate descriptive comments throughout (in addition to the comment mentioned above), to describe what your program is doing and how
- neat and proper indentation and formatting, including consistently and correctly aligning braces using any of the bracing styles illustrated in your textbook
- writing clear and readable code
- use of meaningful descriptive variable names
- use of symbolic constants for all "special" values which don't change and which are used in more than one place in your code
- avoiding disallowed C language features as discussed above
- make sure it runs with the sample input/output files given before submitting it, and make sure you test it on others that are different from those given.