Carleton University
Department of Systems and Computer Engineering
SYSC 2006 - Foundations of Imperative Programming - Fall 2013

Lab 3 - Logic Functions

Objective

We've observed that many students struggle when writing functions that use boolean logic; i.e., `if-else` statements and the logical operators `&&` (and), `||` (or), and `!` (not). In this lab, you'll design, code and test some functions use these constructs.

Attendance/Demo

To receive credit for this lab, you must make a reasonable effort to complete the exercises and demonstrate the code you complete.

When you have finished all the exercises, call a TA, who will review the code you wrote. For those who don't finish early, a TA will ask you to demonstrate whatever code you've completed, starting about 30 minutes before the end of the lab period. Finish any exercises that you don't complete by the end of the lab on your own time. Also, you must submit your lab work to cuLearn by the end of the lab period.

General Requirements

For those students who already know C or C++: when coding your solutions, do not use arrays, structs or pointers. They aren't necessary for this lab.

None of the functions you write should produce console input or output; i.e., contain `scanf` and `printf` statements.

You have been provided with file main.c. This file contains incomplete implementations of four functions you have to design and code. It also contains a *test harness* (functions that will test your code) and a main function that calls these test functions. **Do not modify main() or any of the test functions.**

Instructions

1. Create a folder named Lab 3.
2. Launch Pelles C, and create a new project named lab3 inside the Lab 3 folder. The project type must be Win32 Console program (EXE). (If you've forgotten how to do this, review last week's lab.) You should now have a folder named lab3 inside a folder named Lab 3 (check this).
3. Download file main.c from cuLearn. Move this file into your lab3 folder. **You must also add main.c to your project:** from the menu bar, select Project > Add files to project... In the dialogue box, select main.c, then click Open. An icon labelled main.c will appear in the Pelles C project window.
4. Open main.c.

Sample Exercise

A function named `sleep_in` has two parameters:

```
_Bool sleep_in(_Bool weekday, _Bool vacation);
```

Parameter `weekday` is `true` if today is a weekday, and parameter `vacation` is `true` if we are on vacation. We sleep in if it is not a weekday or we're on vacation. The function should return `true` if we sleep in today; otherwise it should return `false`.

Solution: the condition, "we sleep in if is not a weekday" can be coded as: `if (!weekday)...` The condition, "we sleep in if we are on vacation"can be coded as: `if (vacation)...` Since we sleep in if either condition is true, we use the `||` operator to combine the conditions:

```
if (!weekday || vacation) {
      return true;
} else {
      return false;
}
```

Because the "true" clause in the `if` statement has a return statement, we don't need the `else` clause. Our solution becomes:

```
_Bool sleep_in(_Bool weekday, _Bool vacation) {
      if (!weekday || vacation) {
      return true;
      }

      return false;
}
```

It is better programming style to write `vacation` than `vacation == true`. These expressions mean exactly the same thing, so testing a boolean variable for equality with `true` is superfluous. Likewise, writing `!weekday` is better style than writing `weekday == false`.

Function `sleep_in` has already been defined for you in main.c.

- Build the project. It should build without any compilation or linking errors.
- Execute the project. The test harness will run. Read the console output carefully. The output shows when `sleep_in` is about to be called, the values that are being passed as arguments, the result we expect the function to return, the actual value returned by the function, and a summary of how many tests passed and failed.
- The harness will report several errors while it tests the functions you'll complete for Exercises 1 through 7, which is what we'd expect: you haven't started working on these functions.

Exercise 1
We have a loud talking parrot. Write a function named `parrot_trouble` that has two parameters:

```
_Bool parrot_trouble(_Bool talking, int hour);
```

Parameter `talking` is `true` if the parrot is talking, and parameter `hour` is the current hour in the range 0..23. We are in trouble if the parrot is talking and the hour is before 7 or after 20. The function should return `true` if we are in trouble; otherwise it should return `false`.

Exercise 2
Write a function named `max1020` that has two parameters:

```
int max1020(int a, int b);
```

Parameters `a` and `b` are positive integers values. The function should return the larger value that is in the range 10..20 inclusive, or return 0 if neither value is in that range.

Exercise 3
Write a function named `int_max` that has three parameters:

```
int_max(int a, int b, int c);
```

Parameters `a`, `b` and `c` are integers values. The function should return the largest of the three values.

Exercise 4
The squirrels in Palo Alto spend most of the day playing. In particular, they play if the temperature is

between 60 and 90 (inclusive). But, if it is summer, the upper limit is 100 instead of 90.

Write a function named `squirrel_play` that has two parameters:

```
_Bool squirrel_play(int temp, _Bool is_summer);
```

The function should return `true` of the squirrels are playing, otherwise it should return `false`.

Exercise 5
We are having a party with amounts of tea and candy.

Write a function named `tea_party` that has two parameters:

```
int tea_party(int tea, int candy);
```

This function returns the outcome of the party encoded as an integer: 0 means bad, 1 means good, and 2 means great. A party is good (1) if both `tea` and `candy` are at least 5. However, if either `tea` or `candy` is at least double the amount of the other one, the party is great (2). However, in all cases, if either `tea` or `candy` is less than 5, the party is always bad (0).

Exercise 6
Write a function named `lone_sum` that has three integer parameters:

```
int lone_sum(int a, int b, int c);
```

This function returns the sum of parameters `a`, `b` and `c`. However, if one of the parameters is equal to one or both of the other parameters, then the equal values are treated as 0 when the sum is calculated. For example:

- `lone_sum(3, 2, 3)` returns 2 (the two 3's are treated as 0);
- `lone_sum(3, 3, 3)` returns 0 (the three 3's are treated as 0).

Exercise 7
Write a function named `close_far` that has three integer parameters:

```
_Bool close_far(int a, int b, int c);
```

This function returns `true` if one of `b` or `c` is "close" to `a` (differing from `a` by at most 1) and the other is "far" from `a` (differing from `a` and the "close" parameter by 2 or more).

Note: the C standard library has a function that returns the absolute value of an integer argument. The

function prototype is:

```
int abs(int n);
```

This prototype is found in header file stdlib, so you'll have to remember to put

```
#include <stdlib.h>
```

at the start of your program.

Wrap-up

1. Remember to have a TA review and grade your solutions to the exercises before you leave the lab.
2. The next thing you'll do is package the project in a ZIP file (compressed folder). From the menu bar, select Project > ZIP Files... A Save As dialog box will appear. Click Save. Pelles C will create a compressed (zipped) folder named lab3.zip, which will contain copies of the the source code and several other files associated with the project. (The original files will not be removed). The compressed folder will be stored in your project folder (i.e., folder lab3).
3. Log in to cuLearn and click the Submit Lab 3 link. Submit lab3.zip. Instructions for submitting file-based assignments in cuLearn can be found here:

   www5.carleton.ca/culearnsupport/students/

   Click the link Evaluation Tools, then click the link Assignments.

   After you upload the file, remember to click the Send for marking button (See Step 6 of the instructions).

Posted: Sunday, September 22, 2013.