Lab 4 Grading Guidelines last modified by Aaron Bloomfield on August 13, 2013 11:27:15 AM EDT

Lab 4 grading guidelines

Pre-lab

Out of 10 points. This lab part **is** expected to compile.

A single execution run will call all three of their functions (sizeOf(), outputBinary(), and overflow()). A single integer parameter is given to outputBinary(). There are two execution runs: one with 13 as the parameter, one with 2150 as the parameter. Only the outputBinary() result should differ between the two. Below is a sample output from the first execution run (with 13 as the parameter):

```
Size of int: 4
Size of unsigned int: 4
Size of float: 4
Size of double: 8
Size of bool: 1
Size of char: 1
Size of int*: 4
Size of char*: 4
Size of double*: 4

13 in binary is: 1101

Before: 4294967295
After: 0

This occurs because UINT_MAX is, in binary, a sequence of 32 1's
Attempting to add one to this number results in a 1 followed by 32
0's.  Since the 1 is in the 33rd place, it vanishes, leaving the
final result 0.
```

Note that they may print out 13 as a hexadecimal number (0x0d). And they may have leading zero's on the (binary or hexadecimal) number. The number in the second execution run is 2150, which is 100001100110 in binary and 0x866 in hexadecimal. Any of the formats (hex or binary; leading zero's or not) is fine.

- Binary output: 5 total points
  - 3 points: Execution runs
    - 1 point for correct output from each function (if either of their binaryOutput() runs tank, then take off that point). For overflow(), we are looking that they answered the questions posted in that section of the pre-lab
    - -0.5 points: Missing a data type in size of function (or incorrect size)
  - 2 points: Quality of code
    - 1 point: Some thought was put into the binaryOutput() function
    - 1 point: Some thought was put into the overflow() function
- floatingpoint.doc: 5 total points
  - We're looking to see that they understand the process of converting a floating point number to binary and vice-versa. As long as they show their work and their logic is sound give them full credit.
  - Take points off according to the following guidelines:
    - -5 points: No submission, or no work is shown in both conversions, even if they have an answer(Do not take off any additional points if this is the case)
    - -1 point (for each conversion): Their method for conversions seems to be missing a step
    - -1 point (for each conversion): It looks like they got confused along the way and started doing insensible operations.

In-lab

Out of 10 points. This lab part **is** expected to compile.

inlab4.cpp: Out of 2 points.

Check to see that the inlab4.cpp has all the required sections. Grading guidelines:

- 1 point: For having a section of code that declares variables of type bool, char, int, double, int*, and assigns a value to each of them. There should also be a line that prints out their values.
- 1 point: For having a section of code that declares two 1-dimensional arrays and two 2-dimensional arrays, and also assigns each array initial values.

However, not submitting a inlab4.cpp file at all is -5 points.

inlab4.pdf: Out of 8 points.

Size of C++ data types (6 points): take off 1 point for each answer in the table below that is incorrect.

C++ Type	Size in bytes?	Max value? (base 10)	Zero is stored as (in hex)?	One (or 1.0) is stored as (in hex)
int	4	2^31-1 = 2,147,483,647	0x0	0x1
unsigned int	4	2^32-1 = 4,294,967,295	0x0	0x1
float	4	about 3.4 x 10^38	0x0	0x3f800000
double	8	about 1.7 x 10^308	0x0	0x3ff0000000000000
char	1	127	0x30	0x31
bool	1	true or 1	0x0	0x1
			NULL is stored as?	NULL is stored as?
int*	4 or 8	2^32-1 = 4,294,967,295 or 2^64-1 = 1.84 x 10^19	0x0	0x0
char*	4 or 8	2^32-1 = 4,294,967,295 or 2^64-1 = 1.84 x 10^19	0x0	0x0
double*	4 or 8	2^32-1 = 4,294,967,295 or 2^64-1 = 1.84 x 10^19	0x0	0x0

Primitive Arrays in C++ (2 points): while this is mostly all-or-nothing, you can give 1 point if they got it mostly right (i.e. their answer is close), but made a small mistake.

&(IntArray2D[i][j])= &(IntArray[0][0]) + 4j + 20i

Post-lab

Out of 10 points. This lab part **is** expected to compile.

The execution runs for binary bit counter are passing the numbers 13 and 2150 as command line arguments (as two separate execution runs). The input of 13 yields an answer of 3, and the input of 2150 yields an answer of 5.

Binary Bit Counter: Out of 5 points.

The execution runs are listed above. Grading guidelines:

- 2 points: Proper code execution
  - 1 point for each correct output. If they didn't take in command line parameters (and instead waited for program input), they don't get these two points.
- 3 points: Quality of code
  - 3 points: Code uses recursion to count the number (passed in using a command line argument) of 1s in the binary representation of the number passed in.
  - 2 points: Code looks good, but they made a mistake such as incorrectly getting the arguments.
  - 1 points: Almost no work done, or they didn't use recursion or the command line arguments.
  - 0 points: No work done.

Feel free to interpolate between these values.

Radix Worksheet: Out of 5 points.

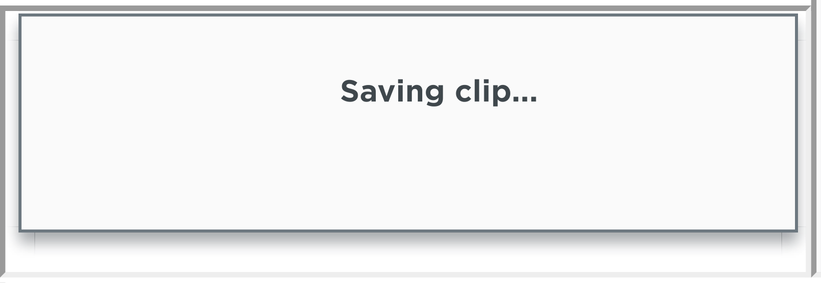
Solutions

Num	Solution
1	47505
2	533
3	3294
4	944
5 a	0x535D
5 b	1010110010100011

Grading guidelines:

- 1 point (for each question, and for question 5, each subpart is 1/2 point): A correct answer

If they show their work and make a mistake, give them as much partial credit as you think they deserve.


Saving clip...