15-213, Fall 20xx
15-213 秋季 20xx

Data Lab: Manipulating Bits
数据实验室: 操作比特

Assigned: Aug. 30, Due: Wed., Sept. 12, 11:59PM
分配时间: 8 月 30 日，预定时间: 9 月 12 日，星期三，晚上 11:59

Harry Bovik (bovik@cs.cmu.edu) is the lead person for this assignment.
Harry Bovik (Bovik@cs.cmu.edu)是这项任务的负责人。

1 Introduction
1 引言

The purpose of this assignment is to become more familiar with bit-level representations of integers and floating point numbers. You'll do this by solving a series of p rogramming "puzzles." Many of these puzzles are quite artificial, but you'll find yourself thinking much m ore about bits in working your way through them.
这个作业的目的是为了更加熟悉整数和浮点数的位级表示。你可以通过解决一系列 p 编程的"谜题"来做到这一点许多这样的谜题都是非常人工的，但是你会发现自己在解决它们的过程中会想到更多关于比特的东西。

2 Logistics
2 物流

This is an individual project. All handins are electronic. Clarifications and corrections will be posted on the course Web page.
这是一个单独的项目，所有的手都是电子的。澄清和更正将张贴在课程的网页上。

3 Handout Instructions
3 份讲义

SITE-SPECIFIC: Insert a paragraph here that explains how the instructor will hand out the `datalab-handout.tar` file to the students.
SITE-SPECIFIC: 在这里插入一段解释讲师如何将 datalab-handout.tar 文件分发给学生。

Start by copying `datalab-handout.tar` to a (protected) directory on a Linux machine in which you plan to do your work. Then give the command
首先，将 datalab-handout.tar 复制到 Linux 机器上的一个(受保护的)目录中，您计划在该目录中完成工作。然后给出命令

```
unix> tar xvf datalab-handout.tar.
Unix > tar xvf datalab-handout.tar.
```

This will cause a number of files to be unpacked in the director y. The only file you will be modifying and turning in is `bits.c`.
这将导致在主管 y 中解压许多文件。你唯一要修改和上交的文件是 bits.c。

The `bits.c` file contains a skeleton for each of the 13 programming puzzle s. Your assignment is to complete each function skeleton using only *straightline* code for the integer puzzles (i.e., no loops or conditionals) and a limited number of C arithmetic and logical operators. Specifically, you are *only* allowed to use the following eight operators:
C 文件包含 13 个编程难题的框架。你的任务是仅使用整数难题的直线代码(即没有循环或条件)和数量有限的 c 算术和逻辑运算符来完成每个函数框架。具体来说，你只能使用以下 8 个操作符:

```
! ~ & ^ | + << >>
!  ~& ^ | + << >>
```

A few of the functions further restrict this list. Also, you are not allowed to use any constants longer than 8 bits. See the comments in `bits.c` for detailed rules and a discussion of the desired coding style.
一些函数进一步限制了这个列表。另外，你不允许使用任何超过 8 位的常量。详细规则和所需编码风格的讨论请参见 bits.c 中的注释。

4 The Puzzles
4 拼图

This section describes the puzzles that you will be solving in `bits.c`.
这一部分描述了你将要解决的字谜。

Table 1 lists the puzzles in rought order of difficulty from ea siest to hardest. The “Rating” field gives the difficulty rating (the number of points) for the puzzle, and t he “Max ops” field gives the maximum number of operators you are allowed to use to implement each function. See the comments in `bits.c` for more details on the desired behavior of the functions. You may also refer to the test functions in `tests.c`. These are used as reference functions to express the correct behavior of your functions, although they don't satisfy the coding rules for your functions.
表 1 按难度大致顺序列出了从最容易到最难的难题。“ Rating”字段给出了难度等级(分数)，“ Max ops”字段给出了允许用于实现每个函数的最大操作符数。查看 bits.c 中的注释，了解更多关于函数的期望行为的细节。你也可以参考 tests.c 中的测试函数。这些函数用作参考函数来表示函数的正确行为，尽管它们不满足函数的编码规则。

Name 名称	Description 描述	Rating 评级	Max ops 最高指挥官
bitXor(x,y) bitXor (x，y)	x ‖ y using only & and ˜. X | | y 只使用 & 和。	1	14
tmin() Tmin ()	Smallest two's complement integer 最小二的补整数	1	4
isTmax(x) isTmax (x)	True only if x x is largest two's comp. integer. 只有当 x x 是最大的 2 的比较整数时，才为 True。	1	10
allOddBits(x) allOddBits (x)	True only if all odd-numbered bits in x set to 1. 仅当 x 中的所有奇数位都设置为 1 时为 True。	2	12
negate(x) 否定(x)	Return -x with using - operator. 带 using-operator 的 Return-x。	2	5
isAsciDigit(x)	True if 0x30 ≤ x ≤.	3	15

isAsciDigit (x) conditional 有条件	如果 0x30≤ x ≤ ，则为 True。 Same as x ? y : z 和 x? y: z 一样	3	16
isLessOrEqual(x, y) isLessOrEqual (x，y)	True if x ≤ y, false otherwise 如果 x ≤ y 为 True，否则为 false	3	24
logicalNeg(x)) 逻辑阴性(x))	Compute !x without using ! operator. 不使用! 运算符而计算! x。	4	12
howManyBits(x) 多少位(x)	Min. no. of bits to represent x in two's comp. 二进制表示 x 的最小位数。	4	90
floatScale2(uf) floatScale2(uf)	Return bit-level equiv. of 2*f for f.p. arg. f. 对于 f.p.arg. f，返回位级等效值为 2 * f。	4	30
floatFloat2Int(uf) floatFloat2Int (uf)	Return bit-level equiv. of (int)f for f.p. arg. f. 返回(int) f 对 f.p.arg. f 的位级等效值。	4	30
floatPower2(x) floatPower2(x)	Return bit-level equiv. of 2.0^x for integer x. 对于整数 x 返回 2.0 x 的位级等效值。	4	30

Table 1: Datalab puzzles. For the floating point puzzles, val ue `f` is the floating-point number having the same bit representation as the unsigned integer `uf`.
表 1: Datalab 谜题。对于浮点数字谜，value f 是浮点数字，其位表示形式与无符号整数 uf 相同。

For the floating-point puzzles, you will implement some comm on single-precision floating-point operations. For these puzzles, you are allowed to use standard control structures (conditionals, loops), and you may use both `int` and `unsigned` data types, including arbitrary unsigned and integer constants. You may not use any unions, structs, or arrays. Most significantly, y ou may not use any floating point data types, operations, or constants. Instead, any floating-point oper and will be passed to the function as having type
对于浮点字谜，您将实现一些通用的单精度浮点操作。对于这些难题，允许使用标准的控制结构(条件、循环)，并且可以同时使用 int 和 unsigned 数据类型，包括任意的 unsigned 和 integer 常量。你不能使用任何联合、结构或数组。最重要的是，你不能使用任何浮点数据类型、操作或常量。相反，任何浮点操作符都会被作为类型传递给函数

unsigned, and any returned floating-point value will be of type unsigned. Your code should perform the bit manipulations that implement the specified floating p oint operations.
任何返回的浮点值都是 unsigned 类型的。你的代码应该执行位操作来实现指定的浮点操作。

The included program fshow helps you understand the structure of floating point numbers . To compile fshow, switch to the handout directory and type:
包含的程序 fshow 帮助您理解浮点数的结构。要编译 fshow，切换到 handout 目录并输入:

```
unix> make
Unix > make
```

You can use fshow to see what an arbitrary pattern represents as a floating-poi nt number:
可以使用 fshow 查看任意模式表示的浮点数是什么:

```
unix> ./fshow 2080374784
/fshow 2080374784

Floating point value 2.658455992e+36
浮点值 2.658455992 e + 36
Bit Representation 0x7c000000, sign = 0, exponent = f8, fraction =
000000 Normalized. 1.0000000000 X 2^(121)
位表示 0x7c000000，符号 = 0，指数 = f8，分数 = 000000 归一化.
1.000000000x2(121)
```

You can also give fshow hexadecimal and floating point values, and it will decipher t heir bit structure.
您也可以给 fshow 十六进制和浮点值，它将破译它们的位结构。

5 Evaluation
5 求值

Your score will be computed out of a maximum of 67 points based on the following distribution:
你的分数将根据以下分布从最高 67 分中计算出来:

36 Correctness points.
正确分数。

26 Performance points.
表现分数。

5 Style points.
风格点数。

Correctness points. The puzzles you must solve have been given a difficulty rating between 1 and 4, such that their weighted sum totals to 36. We will evaluate your functions using the btest program, which is described in the next section. You will get full credit for a puzzle if it passes all of the tests performed by btest, and no credit otherwise.

正确点。你必须解决的难题的难度等级在 1 到 4 之间，所以它们的加权总和为 36。我们将使用最佳测试程序来评估你的函数，这将在下一节描述。如果你的拼图通过了 btest 的所有测试，那么你将获得完全的学分，其他方面没有学分。

Performance points. Our main concern at this point in the course is that you can get the right answer. However, we want to instill in you a sense of keeping things as short and simple as you can. Furthermore, some of the puzzles can be solved by brute force, but we want you to be more clever. Thus, for each function we've established a maximum number of operators that you are allowed to use for each function. This limit is very generous and is designed only to catch egregiously inefficient solutions. You will receive two points for each correct function that satisfies the operator limit.
性能分数。在课程的这一点上，我们主要关心的是你能否得到正确的答案。然而，我们想要灌输给你一种尽可能简短的意识。此外，有些谜题可以用蛮力解决，但我们希望你能更聪明一些。因此，对于每个函数，我们已经建立了一个最大数量的运算符，允许你使用每个函数。这个限制非常慷慨，只是为了捕捉极其低效的解决方案。对于每个满足运算符极限的正确函数，你将得到两分。

Style points. Finally, we've reserved 5 points for a subjective evaluation of the style of your solutions and your commenting. Your solutions should be as clean and straightforward as possible. Your comments should be informative, but they need not be extensive.
风格点。最后，我们为你的解决方案和评论风格的主观评价保留了 5 分。你的解决方案应该尽可能简洁明了。你的评论应该是信息丰富的，但它们不需要是广泛的。

Autograding your work
自动评分你的作品

We have included some autograding tools in the handout directory — `btest`, `dlc`, and `driver.pl` — to help you check the correctness of your work.
我们在讲义目录中包含了一些自动评分工具—— btest、 dlc 和 driver.pl ——以帮助您检查工作的正确性。

- **`btest`:** This program checks the functional correctness of the functions in `bits.c`. To build and use it, type the following two commands:

`Btest`: 这个程序检查 `bits.c` 中函数的功能正确性。要构建和使用它，输入以下两个命令:

```
unix> make
Unix > make
unix> ./btest
Unix > ./btest
```

Notice that you must rebuild `btest` each time you modify your `bits.c` file.
请注意，每次修改 bits.c 文件时都必须重新生成 btest。

You'll find it helpful to work through the functions one at a ti me, testing each one as you go. You can use the `-f` flag to instruct `btest` to test only a single function:
你会发现一次性完成一个函数是很有帮助的，一边做一边测试。你可以使用 -f 标志来指示 btest 只测试一个函数:

```
unix> ./btest -f bitXor
Unix > ./btest-f bitXor
```

You can feed it specific function arguments using the option fl ags `-1`, `-2`, and `-3`:
你可以使用选项 fl ags-1,-2 和-3 来输入特定的函数参数:

```
unix> ./btest -f bitXor -1 4 -2 5
Unix > ./btest-f bitXor-14-25
```

Check the file `README` for documentation on running the `b`test program.
检查 README 文件中关于运行 btest 程序的文档。

- **`dlc`:** This is a modified version of an ANSI C compiler from the MIT CIL K group that you can use to check for compliance with the coding rules for each puzzle. The typical usage is:

`Dlc`: 这是来自 MIT CIL k 组的 ANSI c 编译器的修改版本，您可以使用它来检查是否符合每个字谜的编码规则。典型的用法是:

```
unix> ./dlc bits.c
Unix > 。/dlc bits.c
```

The program runs silently unless it detects a problem, such as an illegal operator, too many operators, or non-straightline code in the integer puzzles. Running with the `-e` switch:

程序静默地运行，除非它检测到一个问题，如非法运算符、太多运算符或整数难题中的非直线代码。使用 -e 开关运行:

```
unix> ./dlc -e bits.c
Unix > ./dlc-e bits.c
```

causes `dlc` to print counts of the number of operators used by each function. Type `./dlc -help` for a list of command line options.
导致 dlc 打印每个函数使用的运算符数量的计数。类型。/dlc-命令行选项列表的帮助。

- **`driver.pl`:** This is a driver program that uses `btest` and `dlc` to compute the correctness and performance points for your solution. It takes no arguments:

Pl：这是一个驱动程序，它使用 `btest` 和 `dlc` 来计算解决方案的正确性和性能点。它不需要参数：

```
unix> ./driver.pl
Unix > ./driver. pl
```

Your instructors will use `driver.pl` to evaluate your solution.
你的导师将使用 driver.pl 来评估你的解决方案。

6 Handin Instructions
6 手册指示

SITE-SPECIFIC: Insert text here that tells each student how to hand in their `bits.c` solution file at your school.
SITE-SPECIFIC: 在这里插入文本，告诉每个学生如何在你的学校提交他们的 bits.c 解决方案文件。

7 Advice
建议

- Don't include the `<stdio.h>` header file in your `bits.c` file, as it confuses `dlc` and results in some non-intuitive error messages. You will still be able to use `printf` in your `bits.c` file for debugging without including the `<stdio.h>` header, although `gcc` will print a warning that you can ignore.

不要在 bits.c 文件中包含 < stdio.h > 头文件，因为它会混淆 dlc 并导致一些非直观的错误消息。您仍然可以在 bits.c 文件中使用 printf 进行调试，而不需要包含 < stdio.h > 头，尽管 gcc 将打印一个您可以忽略的警告。

- The `dlc` program enforces a stricter form of C declarations than is the case for C++ or that is enforced by `gcc`. In particular, any declaration must appear in a block (what you enclose in curly braces) before any statement that is not a declaration. For example, it will complain about the following code:

Dlc 程序强制执行比 c + + 或 gcc 强制执行的更严格的 c 声明形式。特别是，任何声明都必须在不是声明的语句之前出现在一个块中(用大括号括起来的)。例如，它会抱怨下面的代码:

```
int foo(int x)
Int foo (int x)
{
  int a = x;
  Int a = x;
  a *= 3; /* Statement that is not a declaration */ int b = a;
  /* ERROR: Declaration not allowed here */
  A * = 3;/* 不是声明的声明 */int b = a;/* ERROR: Declaration not
  allowed here */
}
```

8 The “Beat the Prof” Contest
8“打败教授”比赛

For fun, we're offering an optional “Beat the Prof” contest t hat allows you to compete with other students and the instructor to develop the most efficient puzzles. The goal is to solve each Data Lab puzzle using the fewest number of operators. Students who match or beat the instructor's operator count for each puzzle are winners!

为了好玩，我们提供了一个可选的“击败教授”竞赛，让你可以与其他学生和教师竞争，开发最有效的拼图。我们的目标是用最少的操作员来解决每个数据实验室的难题。匹配或者击败教练的操作员的学生就是胜利者！

To submit your entry to the contest, type:
要提交参赛作品，请键入:

```
unix> ./driver.pl -u ``Your Nickname''
Unix > ./driver.pl-u“ Your Nickname”
```

Nicknames are limited to 35 characters and can contain alphanumerics, apostrophes, commas, periods, dashes, underscores, and ampersands. You can submit as often as you like. Your most recent submission will appear on a real-time scoreboard, identified only by you r nickname. You can view the scoreboard by pointing your browser at
昵称限制为 35 个字符，可以包含字母数字、撇号、逗号、句号、破折号、下划线和符号。你可以随心所欲地提交。你最近的投稿会出现在一个实时记分板上，只能通过你的昵称来识别。你可以通过将浏览器指向

```
http://$SERVER_NAME:$REQUESTD_PORT
服务器名称: $requestd _ port
```

SITE-SPECIFIC: Replace `$SERVER_NAME` **and** `$REQUESTD_PORT` **with the values you set in the** `./contest/Contest.pm` **file.**
SITE-SPECIFIC: 将 $SERVER _ name 和 $REQUESTD _ port 替换为./contest/Contest.pm 文件中设置的值。