bit操作实验

简介

进一步理解书中第二章《信息的表示和处理》部分的内容，深刻理解整数、浮点数的表示和运算方法，掌握GNU GCC工具集的基本使用方法。

要求

请按照要求补全 bits.c 中的函数，并进行验证。包括以下6个函数：

1) int isAsciiDigit(int x);

- 功能：当0x30<=x<=0x39时（即字符0-9的ASCII码值）返回1；其他情况下返回0
- 示例：
  isAsciiDigit(0x35) == 1
  isAsciiDigit(0x3a) == 0
  isAsciiDigit(0x05) == 0
- 难度：3
- 可使用运算符数：15

2) int anyEvenBit(int x)

- 功能：当x的任意偶数位为1时，返回1；其他情况下返回0
- 示例：
  anyEvenBit(0xA) == 0
  anyEvenBit(0xE) == 1
- 难度：2
- 可使用运算符数：12

3) int copyLSB(int x)

- 功能：将返回值中的所有位全部设置成x中的第0位的值
- 示例：
  copyLSB(5) == 0xFFFFFFFF
  copyLSB(6) == 0x00000000

- 难度：2
- 可使用运算符数：5

4) int leastBitPos(int x)

- 功能：返回一个掩码，在该掩码中标识了二进制数x的最低位编码为1的位置
- 示例：
  leastBitPos(0x60) == 0x20
- 难度：2
- 可使用运算符数：6

5) int divpwr2(int x, int n)

- 功能：计算 x / 2^n，并将结果取整
- 示例：
  divpwr2(15,1) == 7
  divpwr2(-33,4) = -2
- 难度：2
- 可使用运算符数：15

6) int bitCount(int x)

- 功能：计算二进制数x中，对应位值“1”的总位数
- 示例：
  bitCount(5) == 2
  bitCount(7) == 3
- 难度：4
- 可使用运算符数：40

程序内允许使用：

- 运算符： ! ~ & ^ | + << >>
- 范围在0 - 255之间的常数
- 局部变量

程序内禁止以下行为：

- 声明和使用全局变量
- 声明和使用定义宏

- 声明和调用其他的函数
- 类型的强制转换
- 使用许可范围之外的运算符
- 使用控制跳转语句：if else switch do while for

注意：违背以上原则均视为程序不正确！！

评价方法

本次作业总分共30分，其中包括：

- 正确分：18

每个题目都有对应的难度系数，正确完成一道题目则会获得和该题难度系数相同的分值。难度系数总和为15。如果所有的题都做对，则会获得额外3分的加分；反之，没有加分。

- 性能分：12

每到题目都可以使用布尔代数的方法进行暴力求解。但是，我希望大家能够使用一些更聪明和更优雅的方式来解题。因此，我们对每个题目所使用运算符总数进行了限制，如果该题结果正确且运算符总数满足题目要求，则该题获得2分性能分。

- 代码风格分： 3

有意义且清晰的注释（关键在于质量而不是数量）；规范的代码书写格式。

一些说明

操作方法

进入目录/home/simpleedu/lab1，对bits.c文件进行编辑（其他的文件不要进行任何修改），实现文件中所定义的函数。

在作业包中我们还提供了一些辅助工具，协助你提高作业的质量。包括：btest、dlc、BDD checker和driver.pl。

btest

这个程序主要用来检查你在bits.c中所实现的代码的正确性。主要是通过随机生成输入检查你所实现的函数在功能上的正确性。编译并使用btest需要在作业目录下输入以下两条命令：

```
make
./btest
```

每次修改bits.c后，都需要先重新运行make命令才能够使用btest。运行btest可以测试所有的需要实现的函数的正确情况。如果你只需要测试其中的某一个函数，你可以使用-f选项，并制定对应的参数名称来进行测试。例如只测试函数isAsciiDigit的结果，可以输入以下命令：

```
./btest -f isAsciiDigit
```

如果需要测试指定的输入数据我们可以使用选项-1、-2和-3分别制定对应的第一、第二和第三个参数。下面的命令说明，我们测试函数isAsciiDigit，并使用指定的输入参数0x30。

```
./btest -f isAsciiDigit -1 0x30
```

dlc

这个程序主要用于检查我们所编写的代码是否符合作业中所提出的限制条件。可以使用下面的命令来检查：

```
./dlc bits.c
```

使用-e选项可以检查每个函数中的运算符使用情况：

```
./dlc -e bits.c
```

使用-h选项可以查看dlc的其他使用方法：

```
./dlc -h
```

BDD checker

这个工具对你实现的代码进行完备的形式验证。上面提到的btest方法只是通过随机生成的几组数据对你的程序快速的进行随机的测试。这种方法并不能验证在所有的输入情况下，程序都能保证正确的结果。如果需要对你的程序进行完备性的验证，你需要使用BDD checker工具。

如果你需要验证函数isAsciiDigit，是否正确可以在作业目录下执行下面的命令：

```
./bddcheck/check.pl -f isAsciiDigit
```

如果需要验证全部的函数，执行下面的指令：

```
./bddcheck/check.pl
```

使用-g选项可以以表格的形式输出结果：

```
./bddcheck/check.pl -g
```

driver.pl

这个程序同时整合了dlc和BDD checker的功能，将dlc和BDD checker的结果同时输出至控制台。

```
./driver.pl
```

注意：每次修改bits.c文件后都需要在作业目录下首先运行make命令完成编译后，才可以使用上述工具检查你的程序。

实验文件的位置

在simpleedu用户目录的lab1目录下

simpleedu用户密码

Simplexue123

作业的提交

执行make命令成功编译后，会自动生成lab1-handin.zip,将此文件附件上传至系统。不需要填写实验报告。