15213, 20xx 년 가을
데이터 랩: 비트 조작하기
출시일: 8 월 30 일, 마감일: 9월 12 일(수) 오후 11시 59분

Harry Bovik(bovik@cs.cmu.edu) 이 이 과제의 책임자입니다.

1 소개

이 과제의 목적은 정수와 부동소수점 숫자의 비트 수준 표현에 더 익숙해지는 것입니다. 이를 위해 일련 의 프로그래밍 "퍼즐"을 풀어야 합니다. 퍼즐의 대부분은 상당히 인위적인 것이지만, 풀다보면 비트에 대해 훨씬 더 많이 생각하게 될 것입니다.

2 구성

이것은 개별 프로젝트이며 모든 과제는 전자식으로 제출됩니다. 설명 및 수정사항은 수업 웹페이지에 게시됩니다.

3 유인물 지침

여기에 강사가 학생에게 datalab-handout.tar 파일을 배포하는 방법을 설명하는 단락을 넣습니다.

먼저 작업을 수행할 Linux 컴퓨터의 (보호된) 디렉터리에 datalab-handout.tar를 복사합니다. 그런 다음 다음 명령을 실행합니다.

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

이렇게 하면 디렉터리에서 여러 파일이 압축이 풀리게 됩니다. 수정하여 제출할 파일은 bits.c뿐입니다.

bits.c 파일에는 13개의 프로그래밍 퍼즐 각각에 대한 시작 코드가 포함되어 있습니다. 여러분의 과제는 반복문이나 조건문 없이, 제한된 수의 C의 산술 및 논리 연산자만을 사용하여 각 함수를 완성하는 것입니다. 구체적으로 다음 8가지 연산자만 사용할 수 있습니다:

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

일부 함수는 이 목록을 더욱 제한합니다. 또한 8비트보다 긴 상수는 사용할 수 없습니다. 자세한 규칙과 코딩 스타일에 대한 논의는 bits.c의 주석을 참조하세요.

4 퍼즐

이 섹션에서는 bits.c에서 풀게 될 퍼즐에 대해 설명합니다.

표 1에는 가장 쉬운 것부터 가장 어려운 것까지 난이도 순으로 퍼즐이 나열되어 있습니다. "난이도" 항목 에는 퍼즐의 난이도(점수)가 표시되어 있고, "최대 연산자 수" 항목에 각 함수를 구현하는 데 사용할 수 있는 최대 연산자 수가 나와 있습니다. 함수의 구현에 대한 자세한 내용은 bits.c의 주석을 참조하세요. 코딩 규칙을 충족하지는 않지만, 함수의 올바른 동작을 표현하기 위한 참조 함수로 사용되는 테스트 함수는 tests.c에서 볼 수 있습니다.

이름	설명	난이도	최대 연산자 수
bitXor (x, y)	& 및 ~ 만 사용하여 x||y 를 력합니다.	1	14
tmin()	가장 작은 2의 보수 정수를 출력합니다.	1	4
isTmax (x)	x 가 가장 큰 2의 보수 정수인 경우에만 참입니다.	1	10
allOddBits (x)	x 의 모든 홀수번째 비트가 1 로 설정된 경우에만 참입니다.	2	12
negate (x)	$-$ 연산자를 사용하여 $-x$ 를 반환합니다.	2	5
isAsciDigit (x)	0x30 $\leq x \leq$ 0x39 이면 참입니다.	3	15
conditional(x, y, z)	x ? y : z 와 동일합니다.	3	16
isLessOrEqual (x, y)	x $\leq$ y 이면 참, 그렇지 않으면 거짓을 리턴합니다.	3	24
logicalNeg(x))	! 연산자를 사용하지 않고 !x 를 계산합니다.	4	12
howManyBits (x)	x를 2의 보수로 나타내기 위해 필요한 최소 비트 수를 리턴합니다. 니다.	4	90
floatscale2 (uf)	부동소수점 인자 f에 대해 2f를 리턴합니다.	4	30
floatFloat2Int (uf)	부동소수점 인자 f에 대해 (int) f를 리턴합니다.	4	30
floatPower2 (x)	정수 x 에 대해 2.0^x를 반환합니다.	4	30

표 1: 데이터랩 문제. 부동 소수점 퍼즐의 경우, 값 f 는 unsigned int uf 와 동일한 비트를 갖는 부동소수점 숫자입니다.

부동소수점 문제에서는 몇몇 32비트 부동소수점 연산을 구현하게 됩니다. 이 문제들에서는 제어문(조건부, 루프)를 사용할 수 있으며, 임의의 int와 unsigned 타입을 사용할 수 있습니다. 다만 유니온, 구조체 또는 배열은 사용할 수 없습니다. 가장 중요한 것은 부동소수점 타입, 연산 또는 상수를 사용할 수 없다는 것입니다. 대신 모든 부동소수점 숫자는 함수에 unsigned 타입 인자로 전달되며, 반환되는 부동소수점 값도 겉으로는 unsigned 타입입니다. 여러분의 코드는 지정된 부동 소수점 연산을 구현하는 비트 조작을 수행해야 합니다.

포함된 프로그램 fshow는 부동소수점 숫자의 구조를 이해하는 데 도움이 됩니다. fshow를 컴파일하려면 유인물 폴더로 전환하여 다음을 입력합니다:

```
unix> make
```

fshow를 사용하면 임의의 비트 벡터가 부동 소수점 숫자로 표시되는 것을 확인할 수 있습니다:

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

```
Floating point value 2.6584559923+36
Bit Representation 0x7c000000, sign = 0, exponent = f8,
fraction = 000000 Normalized. 1.0000000000 X 2^(121)
```

fshow에 16진수 및 부동소수점 값을 지정하면 해당 비트 구조를 해독할 수도 있습니다.

5 평가

점수는 다음 분포에 따라 최대 67점 만점으로 계산됩니다:

정확도 점수 **36**점

성과 점수 **26**점

스타일 점수 **5**점

정답 점수. 문제들은 각각 1 에서 4 사이의 난이도 등급이 부여되어 있으며, 합이 총 36 이 되도록 되어 있습니다. 다음 섹션에 설명된 btest 프로그램을 사용하여 함수의 정확도를 평가합니다. 여러분의 솔루션이 btest 프로그램이 내는 테스트를 통과하면 점수를 받게 되며, 그렇지 않으면 점수를 받지 못합니다.

수행 점수. 우리가 가장 우려하는 것은 여러분이 정답을 맞히는 것입니다. 하지만 가능한 한 짧고 단순하게 풀어야 한다는 감각도 심어주고 싶습니다. 또한 일부 퍼즐은 브루트 포스로 풀 수 있지만, 더 영리하게 풀 수 있기를 바랍니다. 따라서 각 문제에 대해 사용할 수 있는 최대 연산자 수를 설정했습니다. 이 제한은 매우 관대하며 지독하게 비효율적인 솔루션을 잡기 위해 고안되었습니다. 솔루션이 연산자 제한을 만족할 때 마다 2점을 받게 됩니다.

스타일 점수. 마지막으로, 솔루션의 스타일과 주석에 대한 주관적인 평가를 위해 5 점을 배정했습니다. 솔루션은 가능한 한 깔끔하고 간결해야 합니다. 주석은 유익한 정보를 제공해야 하지만 장황할 필요는 없습니다.

작업물 자동 채점

작업의 정확성을 확인하는 데 도움이 되는 몇 가지 자동 채점 도구(btest, dlc, driver.pl가 유인물 폴더에 포함되어 있습니다.

- **btest:** 이 프로그램은 bits.c에 있는 함수의 정확성을 검사합니다. 이 프로그램을 빌드하여 사용하려면 다음 두 명령을 입력합니다:

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

  bits.c 파일을 수정할 때마다 btest를 다시 빌드해야 한다는 점에 유의하세요.

  각 함수를 한 번에 하나씩 테스트하면서 작업하는 것이 도움이 될 것입니다. -f 플래그를 사용하여 btest에 단일 함수만 테스트하도록 지시할 수 있습니다:

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

옵션 플래그 -1, -2, -3을 사용하여 특정 함수 인수를 전달할 수 있습니다:

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

btest 프로그램 실행에 대한 설명은 README 파일을 확인하세요.

- **dlc:** 이것은 각 솔루션이 코딩 규칙을 준수하는지 확인하는데 사용할 수 있는 MIT CILK 그룹의 ANSI C 컴파일러의 수정 버전입니다. 일반적인 사용법은 다음과 같습니다:

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

프로그램은 허용되지 않은 연산자, 너무 많은 연산자, 허용되지 않은 반복문이나 조건문 등의 문제를 감지하지 않는 한 자동으로 실행됩니다. -e 스위치로 실행하면 각 함수에 사용된 연산자 수를 출력하도록 합니다:

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

./dlc -help 를 입력하면 커맨드라인 옵션을 확인할 수 있습니다.

- **driver.pl:**btest 및 dlc를사용하여 솔루션의 정확도 및 성과 점수를 계산하는 드라이버 프로그램입니다. 인수가 필요하지 않습니다:

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

강사들은 driver.pl을 사용하여 여러분의 솔루션을 평가할 것입니다.

6 과제 제출 지침

여기에 각 학생에게 bit.c를 제출하는 방법을 알려주는 텍스트를 삽입합니다.

7 조언

- <stdio.h> 헤더 파일은 dlc를 혼동하고 직관적이지 않은 오류 메시지를 발생시키므로 bits.c 파일에 포함하지 마세요. <stdio.h> 헤더를 포함하지 않고도 디버깅을 위해 bits.c 파일에 printf를 사용할 수 있습니다. gcc는 무시해도 되는 경고를 출력할 것입니다.
- dlc 프로그램은 C++ 의 경우나 gcc 에서 적용하는 것보다 더 엄격한 형태의 C 선언을 적용합니다. 특히 모든 선언은, 나머지 문장 전에 블록(중괄호) 안에 넣어야 합니다. 예를 들어 다음 코드는 dlc 컴파일러가 썩 좋아하지 않습니다:

```
int foo(int x)
{
    int a = x;
    a *= 3;     /* 선언이 아닌 문장 */
    int b = a;  /* ERROR : 선언은 위 문장 전에 있어야 합니다 */
}
```

8 "교수를 이겨라" 콘테스트

우리는 가장 효율적인 퍼즐을 개발하기 위해 다른 학생 및 강사와 경쟁할 수 있는 "교수를 이겨라" 콘테스트 도전과제를 제공합니다. 여러분의 목표는 참가자 가운데 가장 적은 수의 연산자를 사용하여 각 데이터랩 문제를 푸는 것 입니다. 각 문제에서 강사의 연산자 수보다 같거나 적은 학생들은 우승자가 됩니다! 콘테스트에 응모작을 제출하려면 다음과 같이 입력합니다:

```
unix> ./driver.pl -u "닉네임"
```

닉네임은 35자로 제한되며 영어, 숫자, 아포스트로피, 쉼표, 마침표, 대시, 언더스코어, 앰퍼샌드가 포함될 수 있습니다. 원하는 만큼 자주 제출할 수 있고, 가장 최근 제출물은 실시간 점수판에 표시됩니다. 브라우저에서 다음 링크로 가면 스코어보드를 볼 수 있습니다.

```
http://$SERVER_NAME:$REQUESTED_PORT
```

사이트-특정: `$SERVER_NAME` 및 `$REQUESTED_PORT` 를 `./contest/Contest.pm` 파일에서 강사님이 설정한 값으로 바꿉니다.