

2018 시스템 프로그래밍
- Lab 03 -

제출일자	2018.10.08
분 반	00
이 름	노효근
학 번	201502049

* datalab-handout.tar를 압축해제하고 bits.c 의 함수를 위주로 작성.
(소스코드 캡처 및 설명)

1. bitAnd(int x, int n)

```
int bitAnd(int x, int y) {  
    return ~(~x | ~y);  
}
```

- 드모르간 법칙을 이용하여 구현

2. getByte(int x, int n)

```
int getByte(int x, int n) {  
    return (x >> (n<<3)) &0xff;  
}
```

- n번째 byte 추출 함수
- $n \ll 3$ 이용 하여 왼쪽 3칸 이동 / 8비트 곱
- $x \gg n$ 이용 원한느 바이트를 마지막 8비트로 이동
- 0xff와 x 모두 1인 수 and 하여 원하는값 추출

3. logicalShift(int x, int n)

```
int logicalShift(int x, int n) {  
    return (x>>n) &~(((0x1) << 31) >>n) <<1);  
}
```

- 논리 쉬프트 : 좌측 끝의 MSB에 0을 채워주는 자리이동
- 1을 31만큼 왼쪽으로 이동 32비트 수로 변환, 이흐 n만큼 오른쪽으로 산술 연산 쉬프트 이후 다시 왼쪽으로 논리연산, 값에 NOT을 취한하고 x에 n만큼 오른쪽 쉬프트 한 값을 &연산한다.

3. bitCount(int x)

```
int bitCount(int x) {  
  
    int mask = 0x1;  
    int result;  
    mask = mask | mask << 8;  
    mask = mask | mask << 16;  
  
    result = mask & x;  
  
    result += mask & (x >> 1);  
    result += mask & (x >> 2);  
    result += mask & (x >> 3);  
    result += mask & (x >> 4);  
    result += mask & (x >> 5);  
    result += mask & (x >> 6);  
    result += mask & (x >> 7);  
  
    result += result >> 16;  
    result += result >> 8;  
    result = result & 0xFF;  
  
    return result;  
}
```

- 위 mask 연산을 거치고 나면 비트 패턴이 0000 0001 0000 0001 0000 0001 00001 00001 00001로 만들어 진다. 따라서 8비트 간격으로 1이 있으므로 0~7번 오른쪽 쉬프트 값과 &하고 result 값에 더한다. 이후 result 값에 아래와 같은 연산을 하고 마지막으로 0xff 값을 &하게 된다면 최종 1의 개수가 나온다.

4. isZero(int x)

```
int isZero(int x) {  
    return !(x^0);  
}
```

- x와 0을 XOR 하여 그 값에 Not을 취한다면 x가 0일때만 1을 반환하게 구현하였다.

5. isEqual(int x, int y)

```
int isEqual(int x, int y) {  
    return !(x^y);  
}
```

- x와 y 값을 XOR 하여 값에 Not을 취하여 값이 같은 경우에 1을 반환할 수 있도록 구현하였다.

6. fitsBits(int x, int n)

```
int fitsBits(int x, int n) {  
    int num = 32+~n+1;  
    int temp = (x<<num)>>num;  
    return !(temp^x);  
}
```

- fitsBits(int x, int n) 함수는 n비트로 x의 2의 보수를 표현할 수 있으면 1을 반환하는 함수이다. x가 양수일 경우와 x가 음수일 경우 둘 다 생각해줘야 한다. x가 양수일 경우 n비트로 표현하려면 MSB가 0인 것 까지 고려해서 생각해 봐야 된다.

따라서, 마지막 비트가 존재하는 자리가 m번째 비트라고 하면, m+1로써 수를 표현할 수 있어야 한다.

x가 음수인 경우 n비트로 표현할 수 있다고 생각하면 32비트 중에 32-n은 남는공간으로 가정할 수 있다. 따라서 $((x \ll (32-n)) \gg (32-n)) == x$ 가 성립하면 그 수는 n비트로 표현할 수 있다 -n은 $\sim n+1$ 로 표현한다. x를 $32 + (\sim n)$ 만큼 왼쪽 쉬프트 한 후 다시 오른쪽 쉬프트를 해준다. 그 다음 해당 값이 x와 같은 값인지 확인하기 위해 XOR을 사용한다. 만약 $x^{\text{temp}} == 0$ 이면 $x == \text{temp}$ 라는 뜻이고, $x^{\text{temp}} \neq 0$ 이면 $x \neq \text{temp}$ 라는 뜻이다. 그러므로 $!(x^{\text{temp}})$ 를 반환해주면 된다.

7. isLessOrEqual(int x, int y)

```
int isLessOrEqual(int x, int y) {  
  
    int x_sign = (x>>31)&0x1;  
    int y_sign = (y>>31)&0x1;  
    int check_sign = x_sign^y_sign;  
    int check_1 = check_sign & x_sign;  
    int check_0 = ~check_sign & (((x+(\sim y)) >> 31) & 0x1);  
    return (check_0| check_1);  
}
```

- isLessOrEqual(int x, int y) 함수는 x가 y보다 작거나 같으면 1을 반환하고 아니면 0을 반환하는 함수이다. x와 y의 부호를 각각 받아온 x,y 값을 오른쪽

쉬프트를 31번 한 후(산술 쉬프트), 0x1과

AND연산을 함으로써 선언해주었다. 그리고 x와 y의 부호가 같은지 다른지 알기 위해 같으면 0을, 다르면 1의 값을 출력할 수 있도록 XOR을 사용하여 $x_sign \wedge y_sign$ 을 해준다. 부호가 다른 경우($check_sign == 1$) $x <= y$ 이면 $x_sign = 1$, $y_sign = 0$ 이 된다. 부호가 같은 경우($check_sign == 0$) $x <= y$ 이면 y의 첫 번째 1의 값은 x보다 크다. $x + (\sim y) = 1$ 이므로 $x - y - 1$ 의 부호비트는 1이다.

8. rotateLeft(int x, int n)

```
int rotateLeft(int x, int n) {  
  
    int shift = 33 + ~n;  
    int high = ~0 << shift;  
    int low = ~(~0 << n);  
    int save = ((x & high) >> shift) & low;  
    return (x << n) | save;  
}
```

- rotateLeft(int x, int n) 함수는 x를 n만큼 오른쪽으로 회전하는 함수이다. high, low 변수는 상위, 하위비트를 위한 n비트 공간이다.

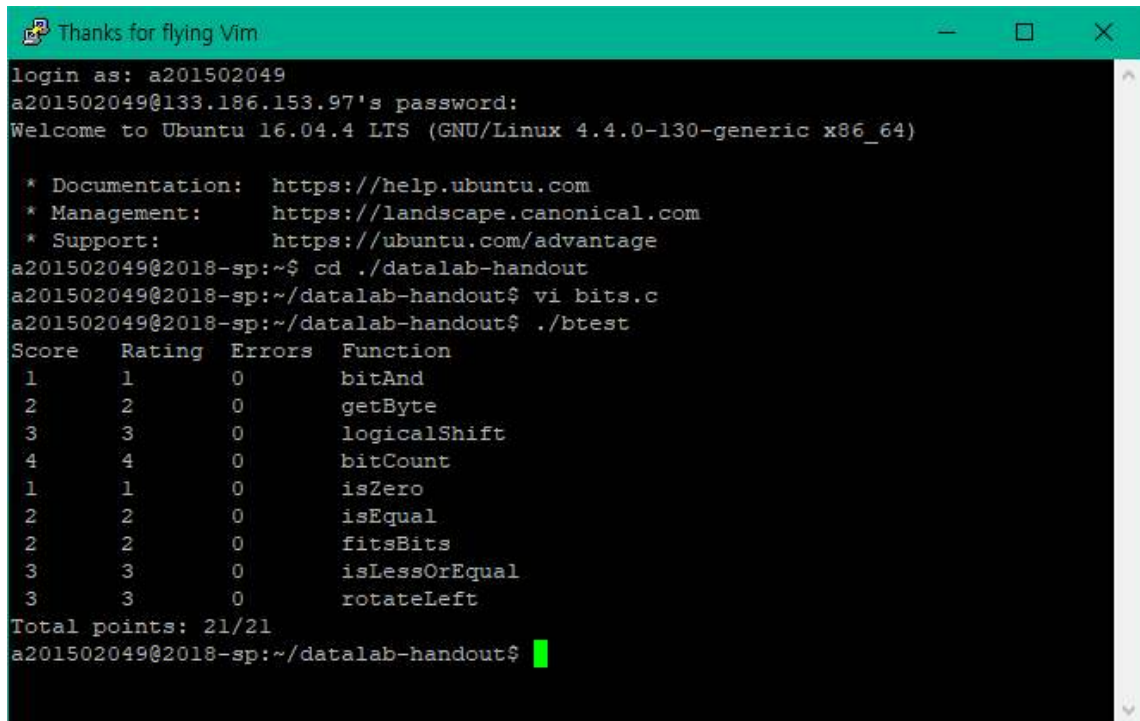
32-n의 수는 $32 + (\sim n + 1)$ 과 같으므로 $33 + (\sim n)$ 의 값을 shift라는 int형 변수에 저장한다. 32-n만큼 왼쪽 쉬프트 연산을 해서 왼쪽에 1이 되게 만든 값을 high에 저장한다.

$\sim(\sim 0 << n)$: n만큼 왼쪽쉬프트 연산 하여 왼쪽 비트가 1이 되게 만든 후 \sim (NOT)을 취해 오른쪽 n비트만큼 1이 되게 한 값을 low에 저장한다.

x와 high값을 AND한 후 32-n만큼 오른쪽 쉬프트 해주면 왼쪽 n비트만큼의 수들이 오른쪽 끝 n개의 자리로 이동된다. 그리고 low와 &연산을 해준 후 save에 저장한다.

x를 n만큼 왼쪽 쉬프트 하게 되면 오른쪽은 n만큼 0이 생기고 이전에 저장해놓은 가장 왼쪽의 n비트를 가장 오른쪽의 n비트로 옮겨놓게 된다.

9. 최종 점수



```
Thanks for flying Vim
login as: a201502049
a201502049@133.186.153.97's password:
Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 4.4.0-130-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage
a201502049@2018-sp:~$ cd ./datalab-handout
a201502049@2018-sp:~/datalab-handout$ vi bits.c
a201502049@2018-sp:~/datalab-handout$ ./btest
Score  Rating  Errors  Function
1      1         0      bitAnd
2      2         0      getByte
3      3         0      logicalShift
4      4         0      bitCount
1      1         0      isZero
2      2         0      isEqual
2      2         0      fitsBits
3      3         0      isLessOrEqual
3      3         0      rotateLeft
Total points: 21/21
a201502049@2018-sp:~/datalab-handout$
```