

비트맵(Bitmap)

[문제] 크기의 2차원 비트맵이 **long long bmap[64]**에 64개의 정수로 표현되어 있다. 우리는 이 **bmap[64]**을 전치시킨(transpose) 새로운 bitmap인 **long long tmap[64]**에 저장하고자 한다. 여러분은 **tmap[64]**에 저장된 64개의 **long long** 정수를 순서대로 출력해야 한다. 아래 표에서 왼쪽은 **bmap[64]={ 19, -21, ... }**, 오른쪽은 **tmap[]**이다.

0	0	0	1	0	0	1	1	bm[0]	0	1	1	0	0	1	0	0	tm[0]
1	1	1	1	0	0	0	0	bm[1]	0	1	0	0	1	1	0	0	tm[1]
1	0	1	0	0	1	1	1	bm[2]	0	1	1	0	0	1	0	1	tm[2]
0	0	0	1	0	0	0	0	bm[3]	1	1	0	1	0	1	0	1	tm[3]
0	1	0	0	1	1	1	1	bm[4]	0	0	0	0	1	1	1	0	tm[4]
1	1	1	1	1	0	0	1	bm[5]	0	0	1	0	1	0	0	0	tm[5]
0	0	0	0	1	0	1	1	bm[6]	1	0	1	0	1	0	1	0	tm[6]
0	0	1	1	0	0	0	1	bm[7]	1	0	1	0	1	1	1	1	tm[7]

[입출력] 입력파일에는 **bm[64]**에 저장된 **long long** 정수 64개가 64개의 줄에 순서대로 주어진다. 여러분은 이를 전치한 **tm[64]** 원소 64개를 순서대로 출력한다. 단 **long long**의 표현 방법은 NESPA에 설치된 표준 GCC 컴파일러에 따른다. 아래는 입출력의 예를 보여주고 있다. 이므로 18자리 이상의 정수 계산에는 **long long**을 사용해서는 안된다. (**C++ long long 자료는 %lld로 처리해야 한다.**)

[예제]

bitmap.inp	bitmap.out
<pre>45LL // 0000000000000000000000000000000101101 -322LL // 1111111111111111111111010111110 73LL // 00000000000000000000000000000001001001 -351LL // 1111111111111111111111010100001 128LL // 000000000000000000000000000000010000000 77LL // 00000000000000000000000000000001001101 -77LL // 11111111111111111111110110011 111LL // 00000000000000000000000000000001101111 ... 45678LL // 64번째 long long bm[63]</pre>	<pre>18345 // tm[0] 4201 // tm[1] -56245 // tm[63]</pre>

[제한조건] 프로그램의 이름은 **bitmap.{c.cpp.py}**이다. 제출횟수는 15회이다. 각 검사 데이터 당 제한시간은 1초이다. 과제 마감시간은 9월 17일(금요일) 저녁9시(21:00)이다. 연습용 데이터가 강의 사이트NESPA에 있으므로 제출 전에 그것으로 확인해볼 수 있다. NESPA

compiler에 대한 질문은 언제든지 질문 게시판으로 가능하다. One-day late rule¹이 적용된다.

¹ One-day late rule: 마감시간 후 24시간 동안 제출이 가능하며 이 때 제출된 과제에 대해서는 10% 감점 한다.