비트맵(Bitmap)

[문제] 64×64 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]
1	1	1	1	0	0	0	0	bm[1]
1	0	1	0	0	1	1	1	bm[2]
0	0	0	1	0	0	0	0	bm[3]
0	1	0	0	1	1	1	1	bm[4]
1	1	1	1	1	0	0	1	bm[5]
0	0	0	0	1	0	1	1	bm[6]
0	0	1	1	0	0	0	1	bm[7]

0	1	1	0	0	1	0	0	tm[0]
0	1	0	0	1	1	0	0	tm[1]
0	1	1	0	0	1	0	1	tm[2]
1	1	0	1	0	1	0	1	tm[3]
0	0	0	0	1	1	1	0	tm[4]
0	0	1	0	1	0	0	0	tm[5]
1	0	1	0	1	0	1	0	tm[6]
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 long의 표현 방법은 ESPA에 설치된 표준 GCC 컴파일러에 따른다. 아래는 입출력의 예를 보여주고 있다. $2^{63} - 1 \approx 9.2233 \cdot 10^{18}$ 이므로 18자리 이상의 정수 계산에는 long long을 사용해서는 안된다. (C++ long long 자료는 %11d로 처리해야한다.)

[예제]

bitmap.inp	bitmap.out		
45LL // 00000000000000000000000000101101	18345 // tm[0]		
-322LL // 111111111111111111111111111010111110 73LL // 00000000000000000000000000001001001	4201 // tm[1]		
-351LL // 111111111111111111111111010100001	••••		
128LL			
-77LL // 11111111111111111111111111111111			
45678LL // 64번째 long long bm[63]	-56245 // tm[63]		

[제한조건] 프로그램의 이름은 <u>bitmap.{c,cpp}</u>이다. 제출횟수는 15회이다. 각 검사 데이터 당 제한시간은 1초이다. 과제 마감시간은 9월 18일(금요일) 자정(12:00 PM)이다. 연습용 데이터가 강의 사이트 ESPA에 있으므로 제출 전에 그것으로 확인해볼 수 있다. ESPA compiler에 대한 질문은

언제든지 질문 게시판으로 가능하다.