문제정의.

Polyomino를 1개에서 최대 5개까지 입력 받을 수 있다. 이 때 각 polyomino의 가로와 세로는 각각 1부터 4까지 가능하다.
이렇게 입력받은 polyomino를 이용하여 정사각형을 만들어야한다.

아이디어.

가능한 정사각형의 한 변의 범위를 정의해보면 다음과 같다.
우선 polyomino들의 가능한 square block(1x1)의 개수는 1개 (n=1, h=1, w=1) ~ 80개 (n=5, h=4, w=4)이다.
정사각형의 한 변의 길이를 k라 할 때 그 넓이는 k^2이고, 8^2 < 80 < 9^2이므로 가능한 정사각형의 한 변의 범위는 $1 \leq k \leq 8$이다.
위의 아이디어를 미루어보아, 블럭의 개수의 제곱근이 정수가 될 때에만 정사각형이 생성될 수 있음을 알 수 있다. 그렇지 않은 경우에는 solution을 만들 수 없다.

따라서 주어진 Polyomino들을 바탕으로 Square Grid의 크기를 정의하고, Polyomino들을 해당 Grid에 서로 겹치지 않고 범위를 넘지 않게 채워넣으며 정사각형을 완성할 수 있는지 체크한다.

- 0으로 초기화된 k x k Grid의 Top-left cell(0, 0)부터 Bottom-right cell(k-1, k-1)까지 (0, 0), (0, 1), (0, k-1), (1, 0), … , (k-1, 0), … , (k-1, k-1)의 순으로 옮겨가면서, 각 polyomino의 square block들을 grid에 채워 나간다.
- 어떤 polyomino를 grid에 채워넣는 것은, 이전 polyomino를 grid에 채우고 난 다음에 남은 cell들에서부터 채워넣는 것으로 시작한다.

해결방법.

1. <입력 및 초기화> polyomino의 개수 n을 입력받고, N만큼 반복하며 각 polyomino에 대한 정보를 입력받는다. 이 때 각 polyomino에는 height, width, shape 그리고 이미 grid에 넣어졌는지를 판단하는 isPlaced를 저장한다. 더불어 입력받은 1의 개수(square block)를 더해가며 총 block 개수를 구한다. polyomino들을 모아 polyominos라는 컨테이너에 저장한다.
2. <Grid 크기 결정> 총 블럭의 개수의 제곱근 k를 구한다. 만약 k가 정수가 아니라면 정사각형을 만들 수 없다. k가 정수라면, k를 변의 길이로 하는 k x k의 squareGrid를 생성하고 0으로 초기화한다.
3. <반복문과 재귀를 이용한 Polyomino 배치> 첫 번째 polyominof를 squareGrid에 배치하는 함수를 호출하는 것으로부터 시작한다. 이 때 해당 polyomino의 id도 함께 넘겨준다.
    a. squareGrid의 첫 번째 cell(0, 0)부터 마지막 cell(k-1, k-1)까지 반복하면서, 해당 cell에 polyomino를 배치할 수 있는지 확인한다.
        i. squareGrid[i][j]의 값이 1이면서 polyomino[x][y]의 값이 1이라면 해당 cell에는 polyomino를 배치할 수 없다. 이 과정을 polyomino의 height, width 크기 만큼 squareGrid의 cell(i, j)에서부터 진행한다.
    b. 배치가 가능하다면 해당 polyomino를 squareGrid에 배치하고, 그 다음 polyomino에 대한 배치를 하는 함수를 호출한다. 이 때,
        i. 만약 현재 배치된 Polyomino가 마지막 polyomino라면 라면 true를 return한다.
    c. 그 다음 polyomino에 대한 배치가 성공하면 true를, 실패하면 false를 받는다.
        i. 만약 그 다음 polyomino 배치가 불가능하다면, 정사각형이 만들어지지 못하는 것이므로, 현재 배치를 취소하고 그 다음 cell로 이동한다. 이 때 배치를 취소하는 것은 해당 polyomino를 채운 squareGrid의 영역을 0으로 만드는 것이다.
4. 모든 polyomino가 성공적으로 배치되면 squreGrid를 출력하고, 그렇지 못하다면 솔루션이 없다는 문구를 출력한다.

시간복잡도.

- 한 polyomino를 k x k 크기의 squareGrid의 각 cell마다 배치 -> O(k^2) - n개의 polyomino에 대해 배치 진행 -> O(n*(k^2)). // single level recursion
- 재귀적으로 호출하며 n개의 Polyomino 배치의 모든 경우를 확인하므로 **O((n*(k^2))^n)**