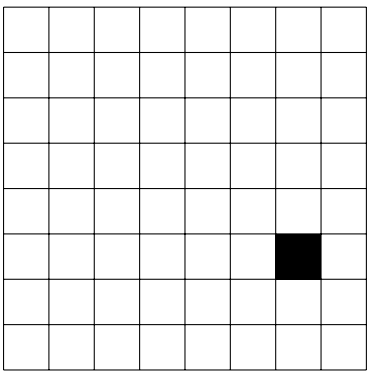
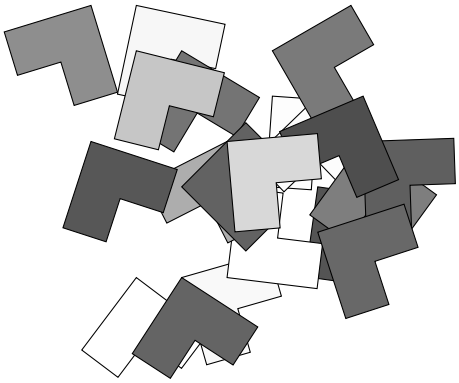
# L-Tiling

## Description

You have a $2^n \times 2^n$ grid, but unfortunately one cell of them is broken.





Now you are using a lot of L-shaped tiles to cover the whole grid. Can you cope with that?

## Implementation

You have to implement the `LTiling` class, and here is the prototype:

```
class LTiling {
  public:
    void tiling(int n, int x, int y);
};
```

Here are some global helper function(s) that you can use:

```
bool put_a_piece(int center_x, int center_y, bool dir_x, bool dir_y);
```

The two boolean variables $dir_x$ and $dir_y$ representing the direction of a L-tile. For example, if a tile is tiled at $dir_x$ =`true` and $dir_y$ =`false`, then it will occupy $(center_x, center_y), (center_{x+1}, center_y), (center_x, center_{y-1})$ these three cells.

The return value of `put_a_piece` is true if you successfully put a piece on the board. The coordinates of the grid is from 0 to $2^n - 1$.

```
void debug(bool colorful);
```

This function is for debugging use. If the variable $colorful$ =`true`, then you'll see a grid with some different symbols representing different tile. If the argument is set to `false`, then you'll see a grid with only hash symbols representing covered spaces.

## Scoring

There is no output, but to get full credit, you have to correctly put every piece that is needed. The order doesn't matter.

## Technical Specifications

- $1 \leq n \leq 10$.
- $0 \leq x, y < 2^n$.

## Sample Input 1

```
2 0 0
```

## Sample Output 1

```
put_a_piece(1, 1, false, false);
put_a_piece(2, 2, false, false);
put_a_piece(3, 3, false, false);
put_a_piece(0, 3, true, false);
put_a_piece(3, 0, false, true);
```