Problem 5-1 (Trominoes) 10 pts

An $n$-tromino is a $2^n \times 2^n$ “chessboard of unit squares with one corner removed” (figure below drawn for $n = 3$). Assume that initially you are given a 1-tromino (i.e., a simple $L$-shaped tile of area 3 drawn on the right), but you have a friendly genie whom you can ask to perform the following two operations in any order:

- **DUPLICATE**: This operation takes an object as input, and creates a second identical copy of this object.
- **GLUE**: This operation takes two objects as input, and glues them together (along the sides, without any overlaps) in a manner specified by you.

(a) (6 points) Design a recursive algorithm $TROMINO(n)$ that creates an $n$-tromino from 1-tromino using the a minimum number of calls to the genie. You only need to specify the top level of the recursion, without the need to explicitly “unwind” the recursion all the way to $n = 1$. *(Hint: First step is to DUPLICATE the original 1-tromino, as otherwise you “lose” it in the recursive call(s), and you might need it in the “conquer” step.)*

(b) (4 points) Give a recurrence relation for the number of calls $T(n)$ to the genie, and solve it.

Problem 5-2 (Coins) 14 pts

Let $n$ be such that $2^{k-1} < n \leq 2^k$. Assume you want to sample a random number $r$ from 1 to $n$.

Consider the following algorithm: sample $k$ random bits, $a_1, \ldots, a_k$. Let $m \in \{0, \ldots, 2^k - 1\}$ be the integer $a_1 \ldots a_k$. If $m < n$, output $m + 1$, else repeat.

(a) (3 pts) Prove that the algorithm is correct; namely, it outputs a random integer $r$ from 1 to $n$.

(b) (4 pts) Let $Z(n)$ be the expected number of random bits used by the algorithm. Write a recurrence equation for $Z(n)$ and show $Z(n) \leq 2k$.

(c) (4 pts) Now, imagine you want to sample a random permutation of the numbers $1 \ldots n$. Start with an array $A$, $A[i] = i$ for all $i$. Consider the following algorithm:
Problem 5-3 (Probabilistic code) 7 pts

Consider the following pseudocode:

\begin{algorithm}
\textbf{Bogus}(n)
\begin{algorithmic}
\State pick random \(x\) in \(\{1, \ldots, n\}\)
\State \textbf{if} \(x == n\) output 1
\State \textbf{else} output \textbf{Bogus}(x)
\end{algorithmic}
\end{algorithm}

(a) (4 pts) Write the recurrence equation for the expected running time \(T(n)\) of the algorithm.

(b) (3 pts) Solve for \(T(n)\).