Problem 1: finding a simple path
Let \( G=(V,E) \) be an undirected graph with \(|V|=n\), and fix \( k>0 \). We saw in class an algorithm based on coloring coding, that checks if \( G \) has a simple path of length \( k \). The runtime of the algorithm was \( \text{poly}(2^k,n) \).

Design a randomized algorithm that finds a simple path of length \( k \) whenever one exists. The runtime should still be \( \text{poly}(2^k,n) \), and the algorithm should succeed with probability at least 50%, say.

Problem 2: finding a bi-partite matching
Let \( G=(U,V,E) \) be a bi-partite graph with \(|U|=|V|=n\). We saw in class an algorithm based on Polynomial Identity Testing (PIT) that decides in poly-time whether \( G \) contains a bi-partite matching or not.

Design a algorithm that (with high probability) finds a bi-partite matching whenever one exists. The runtime should still be \( \text{poly}(n) \), and the algorithm should succeed with probability at least 50%, say.

Problem 3: Reliably and Probably Useful (RPU) algorithms
We proved in class that \( \text{ZPP} = \text{RP} \cap \text{co-RP} \). Here, we will define another model of randomization called RPU (Reliably and Probably Useful), which you will need to prove is also equivalent to ZPP.

An RPU algorithm is a randomized algorithm \( M \), that, given an input \( x \in \{0,1\}^* \), outputs an answer \( M(x) \in \{0,1,?\} \). Here \( ? \) means “I don’t know”. It computes a language \( L \subseteq \{0,1\}^* \) if:

1. It is reliable: when the algorithm makes a prediction (outputs 0 or 1) it has to be correct. Namely, if \( x \in L \) then \( \Pr[M(x) = 0] = 0 \) and if \( x \notin L \) then \( \Pr[M(x) = 1] = 0 \).
2. It is useful: it makes a prediction with some probability on each input. Concretely, for any input \( x \), \( \Pr[M(x) =?] \leq 1/2 \).

Prove that the class of languages that can be computed by an RPU algorithm running in poly-time is the same as ZPP.
Problem 4: PSPACE does not have fixed polynomial size circuits

Recall that

- \( PSPACE = \bigcup_{k \geq 1} SPACE(n^k) \) is the class of languages computable in polynomial space
- \( P/poly = \bigcup_{k \geq 1} SIZE(n^k) \) is the class of languages computable by polynomial size circuits

We believe that PSPACE is not a subset of P/poly, but this is open. In this question you will prove a weaker statement: PSPACE is not a subset of \( \text{SIZE}(n^k) \) for any fixed \( k \).

Fix \( k \geq 1 \). Your goal is to construct a language \( L_k \subset \{0,1\}^* \) that satisfies two properties:

(a) \( L_k \) can be decided in PSPACE. In fact, it is decided in \( SPACE(n^t) \) for some \( t = l(k) \).

(b) There exists \( n_0 = n_0(k) \), such that for all \( n > n_0 \) the language \( L_k \cap \{0,1\}^n \) cannot be computed by boolean circuits of size \( n^k \).

Steps:

1. Fix an input length \( n \). Let \( F_n \) be the class of functions \( f: \{0,1\}^n \rightarrow \{0,1\} \) which can be computed by a circuit of size \( n^k \). Prove that \( |F_n| \leq 2^m \) for \( m = O(n^{k+1}) \).

2. Let \( t \geq 1 \) and fix distinct inputs \( x_1, \ldots, x_t \in \{0,1\}^n \). Prove that there exist values \( y_1, \ldots, y_t \in \{0,1\} \) such that the number of functions \( f \in F_n \) that satisfy \( f(x_i) = y_i \) is at most \( 2^{m-t} \).

3. Argue that for \( t = m+1 \), there are inputs \( x_1, \ldots, x_t \in \{0,1\}^n \) and values \( y_1, \ldots, y_t \in \{0,1\} \) such that any function \( f: \{0,1\}^n \rightarrow \{0,1\} \) which satisfies \( f(x_i) = y_i \) must be outside \( F_n \).

4. Prove that given an input length \( n \), you can find such inputs and outputs in space \( \text{poly}(m) \)

5. Complete the proof - describe \( L_k \) and prove its properties.