Design Turing machines $M = (Q, \Sigma, \Gamma, \delta, \text{start}, \text{accept}, \text{reject})$ for each of the following tasks, either by listing the states Q, the tape alphabet Γ, and the transition function δ (in a table), or by drawing the corresponding labeled graph.

Each of these machines uses the input alphabet $\Sigma = \{\mathbf{1}, \#\}$; the tape alphabet Γ can be any superset of $\{\mathbf{1}, \#, \square, \triangleright\}$ where $\square$ is the blank symbol and $\triangleright$ is a special symbol marking the left end of the tape. Each machine should reject any input not in the form specified below.

---

1. On input $\mathbf{1}^n$, for any non-negative integer n, write $\mathbf{1}^n\#\mathbf{1}^n$ on the tape and accept.

2. On input $\#^n\mathbf{1}^m$, for any non-negative integers m and n, write $\mathbf{1}^m$ on the tape and accept. In other words, delete all the #s and shift the **1**s to the start of the tape.

3. On input $\#\mathbf{1}^n$, for any non-negative integer n, write $\#\mathbf{1}^{2n}$ on the tape and accept. *[Hint: Modify the Turing machine from problem 1.]*

4. On input $\mathbf{1}^n$, for any non-negative integer n, write $\mathbf{1}^{2^n}$ on the tape and accept. *[Hint: Use the three previous Turing machines as subroutines.]*

Questions to ponder:

- Think of a simple problem for which a 2-tape TM seems to offer much better efficiency than a 1-tape TM. Can you argue that 2-tape machine can be simulated by a 1-tape machine with only a quadratic slow down?

- Can you think about why having more than 2 tapes does not buy a lot of speed up? Can you argue why a k-tape TM can be simulated by a 2-tape TM with a slow down that has only only a poly-logarithmic overhead?

- How many bits does each *word* in your laptop/desktop have? How many bits did a desktop have 10 years ago, 20 years ago and 30 years ago? How does it limit the data you can work with?

- Suppose you want to multiply two n bit integers where $n = 10,000$. How would you write a program for it? What would be the time complexity?

- You may know about cryptography and RSA. The current RSA public key is 512 bits. Can you think of an algorithm to check if a given 512 bit number is a prime number? How many steps will it take?

- How can a RAM model with say 64 bits per word be simulated by a k-tape TM? What would be the slow down?