1. (20 points)
   Consider the circuit given below.
   Delays of gates are indicated in the picture. There are no edge (wire) delays.

   (a) Ignoring whether paths are true or false, what is the longest path in the circuit? What is
   the next longest path?
   Also state the lengths of these paths.

   (b) Is there a statistically sensitizable (SS) path in this circuit? (If so, state the path.)
   Explain your answer.

   (c) Is there a statistically co-sensitizable (SC) path that is not statically sensitizable? (If so,
   state the path.)
   Explain your answer.

   (d) Which is the longest true path in the above circuit? Why, and what is its length?

2. (40 points)
   Using the stub code given as part of this assignment, implement efficient polynomial-time
   algorithms for the problems listed below. Ignore issues of true/false paths for this question.
   Test cases are supplied with the stub code, including the simple circuit used in Problem 1.
(a) Find the longest path in a circuit (DAG).
You can implement the algorithm covered in class. If you use a different algorithm, you
must state it clearly and prove its correctness and polynomial-time complexity.
A longest path algorithm is implemented in the Perl Graph module, you can use this for
comparison.

(b) Find the $k$ longest paths in a circuit, where $k$ is an input to the program.
You can implement the algorithm covered in class. If you use a different algorithm, you
must state it clearly (using precise pseudo-code) and prove its correctness (see below*)
and polynomial-time complexity.

(c) Give a polynomial-time algorithm to count the total number of paths in a DAG. Prove
its correctness and running time. Implement this algorithm as a function in the provided
stub code.

(d) Give an algorithm to find all paths in a circuit whose lengths are within $\epsilon$ of the longest,
where $\epsilon$ is an input parameter. Prove its correctness and state conditions under which
your algorithm will run in polynomial time. Implement the algorithm as a function in
the provided code.

Clearly document your code. It should conform to the input-output interface described by
Wenchao during the Sep. 7 discussion.

* Aside: By “prove its correctness”, we mean that you should give a mathematical argument for why your algo-
   rithm is correct. It need not be an extremely detailed proof, but it should include all the key insights.
For example, here is a “proof” for why the CPM algorithm given in class for finding the longest path in a DAG is
correct:

- Topological sort ensures that all predecessors of a vertex $v$ are visited before $v$ is visited.
- We wish to compute the longest distance from source $s$ to sink $f$, $d(f)$. $d(s) = 0$ and for any vertex $v$,
  $d(v)$ is given by

  $$d(v) = \max_{\text{all predecessors } u \text{ of } v} [d(u) + w(u, v)]$$

  Due to the topological sort, the algorithm computes $d(u)$ for any predecessor $u$ of $v$ before computing
  $d(v)$.
  (to make this argument really rigorous, you can use a proof by induction, but that additional detail need not
  be provided)

3. (5 Bonus points just for answering this question)
Describe the top three interesting technical insights you gained about system design, modeling
and analysis from the reading assigned for the first lecture (the IEEE Solid State Circuits
special issue).