Consider the following dag representing a multithreaded computation, where each circle denotes a serially executing strand that takes unit time to execute:

Please provide a numerical answer to the following questions.

- What is the work of this computation?
- What is the span of this computation?
- What is the parallelism of this computation?

Solution:
Work = count the number of nodes. 26
Span = count the length of the longest path. 10
Parallelism = Work / Span = 2.6
Five students have implemented recursive Fibonacci programs, where the base case of each program returns 1 if the program input is \( n = 0 \) or \( n = 1 \). For \( n > 1 \), the various students calculate Fibonacci using the code snippets for the recursive cases shown below:

a: 
\[
\begin{align*}
x &= \text{fib}(n - 1); \\
y &= \text{fib}(n - 2);
\end{align*}
\]

b: 
\[
\begin{align*}
x &= \text{cilk_spawn fib}(n - 1); \\
y &= \text{cilk_spawn fib}(n - 2); \\
\text{cilk_sync};
\end{align*}
\]

c: 
\[
\begin{align*}
x &= \text{fib}(n - 1); \\
y &= \text{cilk_spawn fib}(n - 2); \\
\text{cilk_sync};
\end{align*}
\]

d: 
\[
\begin{align*}
y &= \text{cilk_spawn fib}(n - 2); \\
x &= \text{fib}(n - 1); \\
\text{cilk_sync};
\end{align*}
\]

e: 
\[
\begin{align*}
x &= \text{cilk_spawn fib}(n - 1); \\
y &= \text{fib}(n - 2); \\
\text{cilk_sync};
\end{align*}
\]

Assume that the overhead of spawning a function is about 10 times the cost of an ordinary function call. Rank these codes in order of the performance you would expect for large \( n \). (e.g., fastest \( > \) second fastest \( > \cdots \) slowest):

**Solution:**

d \( \sim \) e \( > \) b \( > \) a \( > \) c

(c) still does everything in serial but has the added overhead of a spawn

(a) is serial

(b) does it in parallel but has the unneeded overhead of two spawns

(d) and (e) introduce the same number of spawns.