1) \( P \): The set of decision problems that can be solved by a Turing machine in polynomial time w.r.t. input.

\( NP \): The set of decision problems whose candidate answers can be verified in polynomial time.

\( NP\text{-}Hard \): The set of decision problems that are as hard as any problem in \( NP \) (i.e., any problem in \( NP \) can be reduced to an instance of an \( NP\text{-}Hard \) problem).

\( NP\text{-}complete \): The set of decision problems which are in \( NP \) and are \( NP\text{-}Hard \).

* Dot product is in \( P \) and \( NP \), as well as \( NC \).
* Internal computations is \( NP\text{-}Hard \) if \( P = NP \).
2) $F = (\neg x_1 \lor x_2) \land (\neg x_1 \lor \neg x_2) \land (\neg x_1 \lor \neg x_2 \lor x_3)$

3 coloring

4 coloring: Just add one more node (colored) and connect all previous nodes to it.
Interval Computation

\[ F(x_1, x_2, x_3) = (1 - (x_1)(1 - x_2))(1 - (x_1)(x_2))(1 - (x_1)(x_2)(1 - x_3)) \]
a. Yes, it belongs to NC (Nick's class) because we can solve the problem in polylogarithmic time on a parallel computer with a polynomial number of processors. This is not an explanation of definition.

b. Is NC = P? Yes. We just simulate all polylogarithmic parallel computations sequentially, and this is still done in polynomial time.

c. Is NC = P? We don't know, but if we can effectively parallelize a P-hard problem such as linear programming and solve it in polylogarithmic time using a polynomial number of processors, then NC = P.

1. If we parallelize and take into account communication time, what is the fastest we get?

   Taking into account communication time:

   \[ R = c \cdot \frac{1}{T \text{par}(n)} \]
   \[ V = \frac{4}{3} \pi R^3 = \frac{4}{3} \pi \left( \frac{3}{T \text{par}(n)} \right)^3 \]
   \[ N \text{proc} \leq \frac{V}{2V} = \left( \frac{4}{3} \cdot \frac{\pi}{\Delta V} \right) \left( \frac{3}{T \text{par}(n)} \right)^3 \]

   \[ T_{\text{seq}}(n) \leq \text{Total time} = T_{\text{par}}(n) \cdot N \text{proc} \leq \text{const} + T_{\text{par}}(n) \cdot T^3 \text{par} \]

   \[ (1 - \frac{7}{16} \leq T_{\text{seq}}(n)) \]

2. Similar arguments are used in the proof that SAT is NP-hard, where \( V \leq C \) and go.

3. The sphere's max volume is bounded by Euclidean space. 
*Explain the physics...

The maximum speed we can travel by is the constant $c$, the speed of light. If we were to travel with no bonds on speed, we could just go as fast as we want. Another scenario would be to effectively travel in time as in a time machine, and send signals to our time when needed.

The volume and the number of processors we can fit inside the sphere, is limited by Euclidean physics. If we were to violate this, and use Lobachevsky's space, we could fit an exponential number of processors in the sphere, and just run many more instances in parallel.

Almost Black hole: Consider a set. The answer is sent back in linear time! The depth of the tree is proportional to $n$ (problem size).

10) $P_{error} = \frac{1}{4}$

\[
\begin{align*}
\epsilon &= 0.1 \\
4^k \leq 10^{-3} \\
4^k \geq 10^3 \\
4^k \geq 1000 \\
(2^{2k}) \geq 1000
\end{align*}
\]
11) \[ \prod P \equiv P = \forall x \exists y ( a + b + c + d \ P ) \text{ combine} \]

12) \[ \exists x \forall y \exists z \text{ win}(x,y,z) \equiv \prod P \]

13) for(int i=0; i<2014; ii++)
    System.out.println("100");
\[ k(x) \leq 46 \]