1. A \( A \cap B \) where \( A \) and \( B \) are r.e. is r.e.
   
   **Proof:** Assuming that it is ok to repeat, the computer can print the elements from \( A \) for 1 hour and then the elements from \( B \) for 1 hr, search for the elements in common and print them as \( A \cap B \), repeat this until finished.

2. A is r.e.
   B is decidable
   
   \( A \cap B \) always r.e.? 
   
   **Yes.** By definition, \( B \), which is decidable, is also r.e. Since \( A \) and \( B \) are r.e. then by the proof on 1, \( A \cap B \) is r.e.

   \( A \cap B \) always decidable?
   
   **No.** The set \( H \) (halt checker) is r.e. but not decidable. If we have \( B = \mathbb{N} \), then \( A \cap B = H \cap \mathbb{N} = H \) \( \uparrow \). Since \( H \) is r.e., but not decidable, we have shown that it is not always decidable (\( A \cap B \) with r.e. and dec.)

3. We can prove this by assuming that this is possible and we call this \( T \)-check.
   To prove that it is not possible, we construct a zero-checker, which we know is impossible, because with a zero-checker we could construct a halt-checker which is also impossible as shown in class.

   ![Zero-checker diagram](image)

   1. public static int q(int p)
      
      return p(n)+n+1 

      p=0 \iff p(n)=n+1 

      zero-checker \( q(n) \) = \( T \)-checker \( q(n)+n+1 \)

      Case 1 \( \forall n (p(n) \leq 0) \):
      
      \( q(n)=p(n)+n+1 \rightarrow \) then
      
      \( q(n)=p(n)+n+1 \rightarrow \) then
      
      Case 2 \( \exists n (p(n)>0) \):
      
      \( q(n)=p(n)+n+1 \rightarrow \) then

      \( q(n)=p(n)+n+1 \rightarrow \) then
Therefore, we have shown that it is not possible to have a program that always checks if $O(n) = n + 1$, because then we could build a zero and halt checker which are impossible.

\[ O_1^2 \text{ and } O \]

\[ O_1^2: \]

\[
\begin{align*}
(\text{start}, 1, #) &\rightarrow (\text{inRight}, 1, R) \\
(\text{inRight}, 1, 1) &\rightarrow (\text{inRight}, 1, R) \\
(\text{inRight}, 1, #) &\rightarrow (\text{inRight}, 2, R) \\
(\text{inRight}, 2, 1) &\rightarrow (\text{inRight}, 2, #, R) \\
(\text{inRight}, 2, #) &\rightarrow (\text{goBack}, 1, L) \\
(\text{goBack}, 1, 1) &\rightarrow (\text{goBack}, 2, L) \\
(\text{goBack}, 1, 1) &\rightarrow (\text{goBack}, 1, L) \\
(\text{goBack}, 2, 1) &\rightarrow (\text{goBack}, 2, L) \\
(\text{goBack}, 2, #) &\rightarrow (\text{start})
\end{align*}
\]

\[ \boxed{\text{halt} 1} \]

\[ O: \]

\[
\begin{align*}
(\text{start}, 2, #) &\rightarrow (\text{goRight}, R) \\
(\text{goRight}, 1) &\rightarrow (\text{goRight}, R) \\
(\text{goRight}, 1) &\rightarrow (\text{return}, 1, L) \\
(\text{return}, 1) &\rightarrow (\text{return}, L) \\
(\text{return}, #) &\rightarrow (\text{halt})
\end{align*}
\]

The third TM is created just by replacing \text{start} with \text{start} 2.

Trace (1, 1)

\[ # \ 1 \ # \ 1 \]

\[ \text{start} 1 \]

\[ # \ 1 \ # \ 1 \]

\[ \text{inRight} 1 \]

\[ # \ 1 \ # \ 1 \]

\[ \text{inRight} 2 \]

\[ # \ 1 \ # \ 1 \]

\[ \text{inRight} 2 \]

\[ # \ 1 \ # \ 1 \]

\[ \text{goBack} 1 \]

\[ # \ 1 \ # \]

\[ \text{goRight} \]

\[ # \ 1 \ # \]

\[ \text{return} \]

\[ # \ 1 \ # \]

\[ \text{return} \]

\[ # \ 1 \ # \]

\[ \text{halt} \]

\[ \sqrt{2} \]
(6) \[ S = 0 \]
for \( \text{int } i = 1; i <= a; i++ \) 
\[ S = S - b \]

\[ S(0, b) = 0 \]
\[ S(m+1, b) = S(m) - b \]
\[ f(0, n) = 0 = g(n) = 0 \]
\[ f(m+1, n) = f(m) - n \]
\[ f(m+1, n) = h(m, n, f(m, n)) = \text{minus}(\frac{m^3}{3}, \frac{n^3}{2}) \]

Then, \[ S = PR(0, \text{minus}(\frac{m^3}{3}, \frac{n^3}{2})) \]

When \( b = 2 \) and \( a = 3 \)

\[ \begin{array}{c|ccc}
   i & S & a & b \\
   \hline
   0 & 0 & 3 & 2 \\
   1 & -2 & 3 & 2 \\
   2 & -4 & 3 & 2 \\
   3 & -6 & 3 & 2 \\
\end{array} \]

\[ S = -6 \]

(7) From (6), we already have a PR expression for \[ S = PR(0, \text{minus}(\frac{m^3}{3}, \frac{n^3}{2})) \]
We obtain \[ \mu_m \text{ as: } \mu_m(S < a) \]
\[ d = \mu_m(S < a) \]

Therefore, \( F(b, a) = S(d, b) \) where \( d = \mu_m(S < a) \) and
\[ S = PR(0, \text{minus}(\frac{m^3}{3}, \frac{n^3}{2})) \]

\[ \begin{array}{c|cc}
   b & a & 3 \\
   \hline
   0 & 3 & 2 \\
   -2 & 3 & 2 \\
   -4 & 3 & 2 \\
\end{array} \]

(8) We can only find an upper bound for the Kolmogorov complexity.
for \( \text{int } i = 1; i <= 2015; i++ \) 
\[ \text{System.out.print("10");} \]
\[ 48 \text{ characters} \]
Therefore \[ K_e(x) \leq 48 \]
Step 1: Truth table assuming $a$ and $b$ are binary 1 digit values:

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>F</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Step 2: Get DNF form for $T_F$

$T_F = (\neg a \& \neg b) V (\neg a \& b) V (a \& b)$

With De Morgan we obtain $F$ in CNF form

$F = \neg \left[ (\neg a \& \neg b) V (\neg a \& b) V (a \& b) \right]$

CNF form: $F = (a \lor b) \& (a \lor \neg b) \& (\neg a \lor b)$

10. $P$ is a problem that can be solved in polynomial time: $\exists P(n) \forall n; t(n) \leq P(n)$

NP are non-deterministic polynomial problems for which once we have a guess we have a feasible algorithm $C(x,y)$ to know if the answer is correct in polynomial time.

NP-hard is a problem harder than all NP problems. Any NP problem can be reduced to an NP-hard problem.

NP-complete is an NP problem that is also NP-hard.

**Sorting** takes polynomial time so it is $P$. It can also be represented as a guess of the correct order so it is also NP. If $P \neq NP$, then this problem is not NP-hard or NP-complete.

Prop SAT

It is NP and NP-hard, because all NP problems can be reduced to it. Therefore it is also NP-complete. There is no proof that $P \neq NP$ therefore if $P = NP$ then SAT is also $P$, but if $P \neq NP$ SAT is not $P$. 

(4)