1. Below you see a Kripke structure; let’s call it $M$. Give its explicit definition in terms of a tuple etc (see the formal definition in the slides).

\begin{center}
\begin{tikzpicture}
  \node (a) at (0,0) {$\{ \text{stable, } x=0 \}$};
  \node (b) at (2,0) {$\{ \text{busy, } x>0 \}$};
  \node (c) at (1,-2) {$\{ \text{busy, } x<0 \}$};
  \node (d) at (0,-2) {$\{ x=0 \}$};
  \path (a) edge[->] (b);
  \path (a) edge[->] (c);
  \path (c) edge[->] (d);
  \path (b) edge[->] (d);
\end{tikzpicture}
\end{center}

(a) Why don’t we have final states there?
(b) How is the notion of ‘execution’ defined for a Kripke structure? And what is an ‛abstract execution’?
(c) Give an execution of $M$ that satisfies the property $X (\text{busy } U (x=0))$. Does $M$ satisfies the property?
(d) So, given a property Kripke structure $M$, an (abstract) execution $\Pi$, and a property $\phi$, and an natural number $i$, what is the difference between:
   - $M \models \psi$
   - $\Pi \models \psi$
   - $\Pi, i \models \psi$

2. Express the following requirements in LTL. Make the necessary assumptions if you have to; but be reasonable.

(a) $P$ and $Q$ cannot not use a resource $r$ simultaneously.
   \textbf{Answer:}
   \[ \Box \neg (\text{use}(P,r) \land \text{use}(Q,r)) \]
   where $\text{use}(P,r)$ is a predicate which is true while and as long as $P$ is using $r$. Importantly note that it does not represent a program call.
(b) If $P$ requests access to $r$, eventually it will get the access.
   \textbf{Answer:}
   \[ \Box (\text{req}(P,r) \rightarrow \Diamond \text{use}(P,r)) \]
   where $\text{req}(P,r)$ is a predicate which is true while and as long as $P$ is requesting for $r$. 
(c) If \( P \) requests access to \( r \), eventually it will get the access; but only if \( P \) persists on maintaining the request.

**Answer:**
\[
\Box(req(P, r) \rightarrow (req(P, r) \lor use(P, r)))
\]

(d) \( P \) cannot access \( r \) without first requesting it; and it cannot do so (make a request) without first releasing \( r \) (if it was busy using \( r \)).

**Answer:**
\[
\Box((\neg req(P, r) \land \neg use(P, r)) \lor (req(P, r) \land \neg use(P, r)))
\]
\[
\Box((use(P, r) \land \neg req(P, r)) \lor (\neg use(P, r) \land \neg req(P, r)))
\]

3. Construct Buchi automata representing the following LTL formulas:

(a) \( p \land q \), where \( p, q \) are atomic propositions.

**Answer:**

(b) \( \neg(x>0 \land x=y) \)

**Answer:** Note that \( \neg(p \land q) = (p \land \neg q) \land (\neg p \land q) \). So the above property is equivalent to:
\[
(x>0 \land \neg x=y) \lor (\neg x>0 \land x=y)
\]

This results in the following Buchi automaton:

(c) \( p \lor (q \land r) \), where \( p, q \) are atomic propositions.

**Answer:**

(d) \( (X x>0) \land x=y \)

**Answer:**

(e) \( \Box \Diamond(x>0 \rightarrow x=y) \)

**Answer:**

(f) \( (p \lor q) \land r \)

**Answer:** I leave this for you.