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).

![Kripke Structure Diagram]

(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 $\Box (\text{busy} \ U \ (x=0))$. 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.
   **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.
   **Answer:**
   $$\Box (\text{req}(P,r) \to \Diamond \text{use}(P,r))$$
   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:**

$$\Box (\text{req}(P, r) \rightarrow (\text{req}(P, r) \lor \text{use}(P, r)))$$

(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:**

$$\Box (\neg \text{req}(P, r) \land \neg \text{use}(P, r)) \lor (\text{req}(P, r) \land \neg \text{use}(P, r))$$

$$\Box (\text{use}(P, r) \land \neg \text{req}(P, r)) \lor (\neg \text{use}(P, r) \land \neg \text{req}(P, r))$$

3. Construct Buchi automata representing the following LTL formulas:

(a) $p W q$, where $p,q$ are atomic propositions.

**Answer:**

```
  p ∈     q ∈ *
  ◯        ◯
```

(b) $\neg(x>0 U x=y)$

**Answer:** Note that $\neg(p U q) = (p \land \neg q) W (\neg p \land \neg q)$. So the above property is equivalent to:

$$(x>0 \land \neg x=y) W (\neg x>0 \land \neg x=y)$$

This results in the following Buchi automaton:

```
  x>0, x=y ¬ ∈  x>0, x=y ¬ ∈ *
  ◯        ◯
```

(c) $p U (q U r)$, where $p,q$ are atomic propositions.

**Answer:**

```
  p ∈              r ∈ *
  ◯                ◯
```

(d) $(X x>0) U x=y$

**Answer:**

```
  x=y ∈           x>0, x=y ∈ *
  ◯                ◯
```

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

**Answer:**

```
  x=y ∈           x>0 ∈
  ◯                ◯
```

(f) $(p U q) W r$

**Answer:**

```
  x>0 ∈     x=y ∈               x>0 ∈
  ◯         ◯                   ◯
  x>0 ∈     x=y ∈               x>0 ∈
  ◯         ◯                   ◯
```

2
With a single accepting group, namely $F = \{1, 3, 5\}$. Accepting via 1 describes executions whose prefix repeatedly satisfy $p U q$, zero or more times, and ends up in $q$; and then it is followed up with $r$.

Accepting via 3 describes the scenario executions that remain in $p U q$ forever, without ever to go over to $r$.

Finally, accepting via 5 describes executions whose prefix repeatedly satisfy $p U q$, zero or more times, and then they go over to $r$; thus still owing the future $q$, but this future $q$ is met (after $r$).