1. For externel peer, at router E, router D's advertised path is at a higher local preference than router B's (which is at 99 signifying lower priority). Similarly, at router D, router D would prefer C's path since it is advertised with a higher local prederence. (B and E are part of the same path to the external peer at 172.0.0.0/24, both with local preference of 2). At C, the same thing happens in which since acc to our policy router A is the preferred exit route. At router A, there's a regex that basically states that any traffic to the external peer is the default local preference of 100, whereas for router B (the other exit router form our AS100), the same regex gives it a local preference value of 99.

2. For carry-in traffic to router E, router E advertises its subnet of 1.2.3.4/24. At node A, since that's the entry node that's preferred according to our policy, compared to the one advertised by node B which has a prepended value of 2, which explicitly states that the route through B is longer hence not preferred. This leads to the effect at subsequent nodes which cause paths going through A to E to be more preferred than those with entry point being B. That's why at C, we see D's path (which in turn is directly next to E), as being the only allowed path. Otherwise packets are dropped.

3. For the bad gadget BGP AS configuration, upon writing each of the following rules in the policy file -

```
1.0.0.0/24 => path(B,D,A) >> path(B,A) + path(C,B,A) >> path(C,A) + path(D,C,A) >> path(D,A),
  #1.0.0.0/24 => (path(B,D,A) + path(C,B,A) + path(D,C,A)) >> (path(B,A) + path(C,A) + path(D,A)),
  true => drop
```

Leads to a compiler error where it cannot find preferences for router D. This makes sense because from the point of view of Propane, it wants to ensure that a stable convergence arises under all cases. In this case, when it arrives at hop D, it then selects the preference route for it, consisting of router C, which in turn selects A, and the reselects D, thus Propane not being able to converge. I tested with both the rules (meaning the same thing) since I wanted to ensure that the distributive property of >> over + is maintained, and that we get the same error due to this.

Note - Router A corresponds to Router 0, Router B to Router 1, Router C to Router 2, Router D to Router 3 from the question.

-------- Cold Potato Routing --------

```
define main = {
  1.0.0.0/24 => path(B,D,A) >> path(B,A) + path(C,B,A) >> path(C,A) + path(D,C,A) >> path(D,A),
  #1.0.0.0/24 => (path(B,D,A) + path(C,B,A) + path(D,C,A)) >> (path(B,A) + path(C,A) + path(D,A)),
```

```
  true => drop
}
```