Uppgift 1

Svårigheten med uppgift 1 var att förstå själva uppgiften samt den givna koden för att inse att det var i new_value funktionen man behöver anpassa vad värdet ska sättas till beroende på de kända variablerna. Vi löste det genom att kolla vilka av inparametrarna som redan hade värden och därifrån använda korrekt op/inverse_op när värdet angavs för den kvarvarande parametern som inte hade något. Vi fick sedan bygga nätverket med våra multipliers och adder så att de använde sig av de korrekta operatorerna genom att ange inparametrarna på rätt sätt.

Uppgift 2

Vi upplevde att uppgift 2 var mycket svårare än uppgift 1 eftersom det fanns mycket mer given kod. Vi började med att kolla vad debuggern printade för något och insåg då att det behövdes läggas till en funktion som gör att både Multiplier och Adder ska använda sin inverse operator vid division respektive subtraktion. Vi la också till några prints och efter ett tag insåg att vi också behövde lägga till en funktion som kollar om en connection är en connector eller ett constraint (get_connector). Om det är en connection så returnerar vi bara den om det är en arithmetic constraint så returner vi out connectorn.

Övrigt

Det var väldigt rörigt att sätta sig in i koden både eftersom det var svårt att förstå vad man ska göra, men också mycket pga det fanns en hel del nya strukturer och funktioner som är unika för Ruby som vi inte stött på tidigare.

På uppgift ett stötte vi t.ex. på funktionen send som vi först trodde var en del av koden och inte inbyggt i Ruby. Den visar sig dock vara en väldigt kraftfull funktion som skickar ett meddelande till ett objekt som sedan kallar på en metod.
val=a.value.send(op, b.value)

I uppgift två stötte vi tex på [:alpha:] som var något som kallas en POSIX bracket och var ekvivalent till [a-zA-Z]
token(/[[:alpha:]]+/)

Sedan stötte vi även på nedan kodbit som vi var förvirrade över ett tag. Vad vi förstår så kollar den om connector finns i @connectors och om den finns sätts den till sig själv annars skapas en ny connector som läggs till i @connectors.
@connectors[connector] ||= Connector.new(connector)

Summa summarum gjorde kombinationen att försöka förstå constraint networks och parsers tillsammans med flexibiliteten och otydligheten med Ruby uppgiften väldigt svårlöst och frustrerande.