1:
Börja med att ändra ordningen på vyerna så att titlarna soft, medium och hard syns över äggbilderna

2:
Länka upp knapparn till en IBAction och döp den till ex hardnessSelected. Testa sedan att alla fungerar med ett printstatement som skriver ut currentTitle.

Hint: Skapa en variabel som ni döper till hardness som är lika med sender.currentTitle. Printa ut den variabeln.

3:
Skapa ett Dictionary utanför IBAction (ovanför, men inom ViewController som ni döper till eggTimes. I detta skriver ni "Soft": 5, "Medium": 7, "Hard": 12

Om ni nu vill ha tag på värdena i eggTimes så skapa en ny variabel inom IBAction döp till result sätt den lika med eggTimes[hardness]! Tänk på att värdet är en optionals och eftersom vi vet vad vi har för knappar forceUnWrappar vi den med utropstecknet.

4:
Nu måste vi skapa en timer som räknar ner koktiderna och då får vi börja med att konvertera minuterna till sekunder.
5 => 300, 7 => 420, 12 => 720
Där vi har vårt Dictionary på raden under så skapar vi en variabel Som
vi döper till secondsRemaining och tilldela värdet 60

Den variabeln använder vi i vår IBAction så att vi kan få tag på våran hardness dvs: secondsRemaining = eggTimes[hardness]! På så sätt kan vi hålla reda på de olika koktiderna.

Nu behöver vi en Timer som räknar ner till noll och om ni googlat rätt har ni något som börjar Timer.scheduledTimer(en massa argument...

Det argumentet som heter timeIntervall ska ha värdet 1.0, Vi vill att repeat är true, Vidare så måste ni ange i selector vad som ska styra timern (nedräkningsfunction.

Här är en bra förklaring hur Timers kan användas på olika sätt.

https://www.hackingwithswift.com/articles/117/the-ultimate-guide-to-timer

Som ni ser så klagar Xcode om vi bara skriver func updateTimer()
Det är för att argumentet selector är i Objectiv-C men vi skriver swift function. Klicka på varningen Så fixar ni till det genom att använda fix-knappen. Då läggs prefixet @objc till före func
Och Xcode är go o gla.

5:
Nu när vi har satt selector-argumentet till att kalla på funktionen updateTimer() så måste vi skapa innehåll i den.

Med en if som kollar att secondsRemaining är större än 0
Då kan vi
print(”\(secondsRemaining) seconds.”)
secondsRemaining -= 1

Då kör vi appen och ser vi om timern räknar ner.

6:
Nu räknar vår timer ner men den har en bugg. Man kan starta flera timer samtidigt och det gör att den blir förvirrad och löper amok. Det beror på att vi alldrig stänger av timern.

Det fixar vi genom att skapa en ny variabel globalt döp den till timer och tilldela den värdet Timer()

Om vi nu går in i vår IBAction så över all annan kod använder vi timer och med dot-noterin kallar på functionen invalidate()
Det gör att vi stannar timern varje gång innan vi startar en ni timer.

Om ni kör appen så ska den nu inte löpa amok när ni trycker på en ny koktid. Utan den startar om med det värdet som hör till respektive knapp.

7:
Nu så ska ni ändra så att er label ändras till Done! eller Klart! När timern är klar. Börja med att skapa en IBOutlet för er label och döp till typ titleLabel.

När ni gjort det så måste vi lägga till ett else-statement i vår updateTimer()
Där är det ju bra att först invalidate timern på samma sätt ni gör direkt när ni startar en timer.
Nu kan ni via titleLabel komma åt er text-property och sätta den till ”DONE!”

Hint: För att göra testningen enklare så bör ni ändra värdena på koktiderna till: 3, 4 ,7 sekunder annars tar det en evighet att testa er kod.

8:
Nu ska vi lägga till en progressView underst i appen. Om ni kollar i stackView så är där en oanvänd View som heter Timer View. Ta och dra in er progressBar i den View’n.
Markera progressbaren och pinna den till 0 höger och vänster lägg till de constraintsen se även till att lägga till en Vertically constraints så att den är imitten.
Nu om ni fortfarande har progressView i focus så gå till inspectorn och ändra Style från default till Bar. Detta gör att jag kan gå in och ge den en constraints för height och vi sätter den till 5 punkter.
Ändra nu färgen på: Progress Tint till system yellow color
Och Track Tint till system grey eller någon färg ni tycker passar in.

En progressBar funkar som så att den går från 0 till 1. Det sistnämda är att den är 100% full i vårt fall så är hela då gul.
Om den är o så är den i vårt fall grå(eller vad ni valde).

Propertyn heter Progress och den finns i inspektorn där ni kan leka med värdena för att se hur allt funkar.

Nu behöver vi en IBOutlet för vår progressBar. Länka upp den och kalla den för progressBar (var uppmärksamma på att typen är UIProgressView).
Nu kan vi testa att det funkar genom att anropa vår bar

progressBar.progress = 1.0 <= Float = viktigt
Om vi kör appen och trycker på någon knapp så ska hela bli gul (om ni använde den färgen).

9:
Vi vill beräkna koktiden men då behöver vi hålla reda på två variabler. Så för enkelhetens skull byter jag namn på secondsRenaining till: totalTime och sätter den till 0 (tänk på att ändra i er IBAction också.
Vi skapar en ny variabel under totalTime och namnger den till secondsPassed och tilldelar den värdet 0

Nu måste vi ändra variabelnamn i vår updateTimer()
Och vår if-sats ändras till secondsPassed < totalTime
Och istället för att räkna ner så räknar vi upp.
secondsPassed += 1

Nu ska vi använda våra värden för att få fram ett decimalnummer mellan 0 och 1. Så vi dividerar secondsPassed med totalTime
Men vi måste först sätta progress-propertyn till vår outlet som vi namngav progressBar.

Om ni inte får progressbaren att funka så är en hint att tiden måste typas om till Float

Slutligen så för att snygga till appen så måste vi nollställa vår progressBar och secondsPassed till o.o resp. 0 Därtill så kan vi återställa vår titleLabel.text till hardness

10:
Använd nu kunskapen från Xylofonappen i hur man spelar ljud och applicerar en player som spelar alarmljudet när nedräkningen är färdigt.
Hint: Där ni ändrar titleLabel till DONE!