Uppgift 1

Kirk för en loggbok ombord sitt skepp över alla brädspelsträffar han har. Han är inte den skarpaste kniven i lådan och har lite svårt att veta hur lång tid det gått mellan de olika träffarna. Av denna anledning vill han att du skapar ett datorprogram som mäter antalet sekunder mellan två klockslag.

Du kan anta att användaren gör rimlig inmatning av 2 tider på formatet hh:mm:ss. Det är en 24-timmarsklocka som används.

Körexempel 1:

```
Första tiden: 23:59:00
Sista tiden : 00:01:00
Det är 120 sekunder mellan första och andra tiden.
```

Körexempel 2:

```
Första tiden: 12:00:00
Sista tiden : 11:59:59
Det är 86399 sekunder mellan första och andra tiden.
```

Körexempel 3:

```
Första tiden: 01:20:00
Sista tiden : 01:21:30
Det är 90 sekunder mellan första och andra tiden.
```

Tips:
Det är förmodligen smart att bryta ner detta problem i mindre delar. Du skulle exempelvis kunna skapa en funktion för att öka timmarna i klockslaget med 1 och sedan skapa en funktion som räknade upp minuterna med 1 och så vidare.

Uppgift 2

Sisko gillar att spela brädspel. Framförallt är han just nu inne på att spela mycket Descent. Descent är en dungeoncrawler där spelaren väljer en karaktär att spela. Varje karaktär är antingen av typen magiker eller krigare. Beroende på vilken typ karaktären är så kan den välja mellan ett antal olika specialiseringar. I filen specialiseringar.txt så finns en lista av alla specialiseringar i spelet och vilken typ av karaktär som kan välja den specialiseringen.

Ditt jobb är att skapa ett program som läser in filen **specialiseringar.txt** och skriver ut alla möjliga kombinationer av karaktärstyper och specialiseringar. Filen ska kunna innehålla ett godtyckligt antal typer och specialiseringar. Specialiseringarna i filen är strukturerade på följande sätt:

[specialisering] – [typen som kan använda specialiseringen]

specialiseringar.txt innehåller:

```
runsmed – magiker
själsskådare – magiker
kämpe – krigare
bärsärk - krigare
```

Körexempel:

```
magiker:
-------------
      runsmed
 själsskådare

krigare:
-------------
        kämpe
      bärsärk
```

Uppgift 3

Janeway älskar brädspelet Descent Journeys in the dark. I Descent så slåss mäktiga hjältar mot fantastiska monster under tiden de utforskar grottor i världen Terrinoth. Tyvärr tycker hon att det är lite mycket att hålla reda på i detta spel. Framförallt tycker Janeway att det är jobbigt att hålla koll på om monstren tagit tillräckligt med skada för att duka under eller inte.

Din uppgift är att skapa en **ADT** för att hantera monster och skada. Det finns given kod i filen ”**uppgift3.py**”, du skall skapa de funktionerna som krävs för att den givna koden skall fungera. Du ska också välja lämpliga inbyggda datatyper och följa principerna för att designa en bra **ADT**.

Funktionerna du ska skapa:

create_monster: create monster tar namn och hälsa som parametrar och returnerar ett monster.

display_monster: denna funktion tar emot ett monster som parameter och skriver ut informationen på korrekt sätt, se körexemplet. Funktionen har inget returvärde.

decrease_health: tar emot ett monster och ett heltal och minskar monstrets hälsa med lika mycket som heltalet.

is_alive: returnerar sant om monstret har mer än 0 hälsa kvar. Annars returnerar funktionen falskt.

Körexempel:

```
Följande monster skapades:
name: orc
health: 4

Följande monster skapades:
name: dragon
health: 30

monster 1 har förlorat 2 liv,
den ser nu ut så här:
name: orc
health: 2

draken lever: True
monster 1 har förlorat 2 liv till
orken lever: False
```

Uppgift 4

Archer ska spela ett nytt rollspel ombord. Det här rollspelet använder lite speciella tärningar, men eftersom han är lite snål så vill han inte köpa dessa speciella tärningar. Istället ger han dig uppgiften att skapa ett program som kan ersätta tärningarna. Eftersom hans bågskytt (bågskytt… archer… get it?) använder gula och gröna tärningar så är det dessa ditt program ska kunna hantera.

2 olika tärningar:

Färg på tärningen:	Möjliga resultat:
Grön (8 sidor)	A, B, D, G, K, M, P, X
Gul (6 sidor)	1, 3, 5, 7, 9, 11

Programmet frågar hur många tärningar av varje typ användaren vill ”rulla”. En rull innebär att slumpa fram ett resultat från de möjliga resultaten. Sedan ska programmet presentera en lista över varje unikt resultat.

TIPS: *Du kan hantera slump i Python med hjälp av modulen ”random” (se dokumentationen).*

Körexempel:

```
Mata in hur många Gröna tärningar du vill slå: 40
Mata in hur många Gula tärningar du vill slå: 2

D
1
P
G
K
11
A
X
B
M
```

Körexempel:

```
Mata in hur många Gröna tärningar du vill slå: 2
Mata in hur många Gula tärningar du vill slå: 1

M
B
7
```

Uppgift 5

Picard ska spela Descent tillsammans med sina kompisar ombord. En stor del av spelet handlar om att hantera en kortlek (Picard… Picka A Card… get it?) med massor av spännande kort i. I den givna filen uppgift5.py finns det given kod som skapar en kortlek och sorterar denna på olika sätt. Ditt jobb är att implementera funktionen ”sort_deck” så att den fungerar som förväntat med den givna koden.

Det finns 3 data associerade med varje, kort: namn, kostnad och kraft. Det första anropet till sort_deck sorterar leken efter **cost** och det andra anropet sorterar leken efter **power**.

Körexempel:

```
Här skrivs korten ut sorterade på första siffran.
Awesome card 2 7
Awesome card 3 5
Awesome card 4 6

Här skrivs korten ut sorterade på andra siffran.
Awesome card 3 5
Awesome card 4 6
Awesome card 2 7
```