Uppgift 1

Kapten Kirk har en dator i korridoren utanför sitt kontor. Han vill använda denna dator så att besättningen kan se alla framtida brädspelsträffar. Tanken är att en person skall kunna köra ett program på datorn genom terminalen. Detta program skall ha 2 olika funktioner:

1. Lista alla framtida träffar
2. Skapa en ny träff

Du får själv välja hur du sparar denna data men den skall finnas kvar mellan körningar av programmet, du kommer alltså behöva spara datan i en fil. En träff behöver följande information:

1. Namn på spelet
2. Starttid
3. Längd

Körexempel 1 (Programmets output är i fetstil):

```
$ python3 uppgift1.py new Gloomhaven 06:00 140
$ python3 uppgift1.py new Descent 09:00 60
$ python3 uppgift1.py list
Future games
=============
Gloomhaven:
Start: 06:00
Length: 140 minutes

Descent:
Start: 09:00
Length: 60 minutes
$ python3 uppgift1.py new Twilight 12:00 360
$ python3 uppgift1.py list
Future games
=============
Gloomhaven:
Start: 06:00
Length: 140 minutes

Descent:
Start: 09:00
Length: 60 minutes

Twilight:
Start: 12:00
Length: 360 minutes
```

Uppgift 2

Kapten Janeway och hennes vänner köper ofta mat tillsammans när de spelar brädspel. På sistone har det blivit lite besvärligt att hålla reda på vem som är skyldig vem pengar. Därför har hon gett dig i uppgift att skriva ett program som sköter detta. Programmet fungerar så att användaren matar in ett namn och ett belopp separerade med mellanslag. Beloppet kan vara positiv eller negativ. Personens nuvarande skuld adderas sedan med det givna beloppet. Programmet ska fungera för ett godtyckligt antal personer. Programmet ska skriva ut nuvarande skulder när användaren matar in ”skriv ut” och ska köra fram tills användaren matar in ”avsluta”.

Körexempel 1 (användarinmatning är i fetstil):

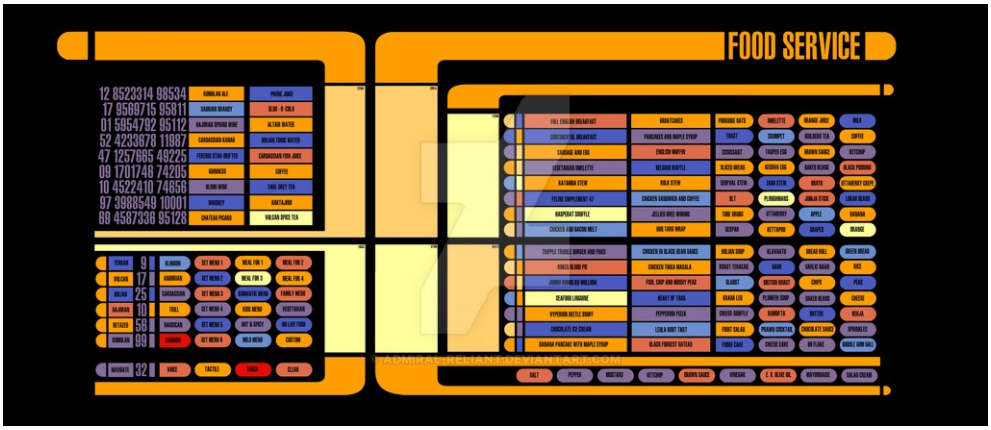
```
Välkommen till skuldprogrammet
Mata in transaktioner:
Kim 40
Alex -40
Sam -20
Kim -20
skriv ut

Skulder:
========
Kim: 20
Alex: -40
Sam: -20

Kim -20
Sam 20
skriv ut

Skulder:
========
Kim: 0
Alex: -40
Sam: 0

avsluta
```



Uppgift 3

Sisko arbetar mycket med datorer och funderar på att anställa dig som besättningsmedlem för att minska sin arbetsbelastning. För att visa att du kan programmera vill han att du skriver ett mindre program och implementerar en sorteringsalgoritm.

Programmet skall slumpa fram 10 tal i intervallet 1 till och med 10 och spara dessa i en lista. Denna lista skall sedan skickas som parameter till en funktion som följer följande specifikation:

- Tar en godtyckligt lång lista som indata
- Har inget returvärde (ändrar på listan som en referens)
- Gör följande med listan för att sortera den:

  För vaje index **i** i listan gör följande:

    Sök alla index **i+1..N** efter det minsta värdet

    Kalla det index med minsta värdet för **i_min**

    Om värdet på index **i_min** är mindre än värdet på index **i**:

      Byt plats på index **i** och **i_min**

Tips:

Det är ok att skriva fler funktioner för att bryta ner problemet i mindre delar. Exempelvis kan det vara lämpligt att skriva en funktion som hittar indexet för det minsta talet.

Du kan använda funktionen **randint** som finns i modulen **random** för att generera slumpade heltal.

Du skall följa specifikationen för funktionen sort som finns ovan, du får inte göra extra saker för att lösa problemet.

Körexempel 1:

```
python3 uppgift3.py
[1, 6, 6, 7, 7, 7, 8, 8, 8, 10]
```

Uppgift 4

Archer håller på att dokumentera sin samling av brädspel. Därför vill han att du skapar en adt för brädspel. I filen uppgift4.py finns given kod som skapar komponenter och brädspel. Ditt jobb är att skapa en adt för ett spel och en adt för en komponent.Varje komponent har ett namn, en vikt och en beskrivning. Varje Spel har ett namn, en beskrivning och består av ett godtyckligt antal komponenter.

create_component – Tar emot namn, vikt (i gram) och en beskrivning av komponenten och returnerar en komponent

print_component – Tar emot en komponent och skriver ut all information om komponenten formaterat enligt körexemplet

get_component_weight – Tar emot en komponent och returnerar komponentens vikt i kg

create_game – Tar emot namn, beskrivning och om användaren vill en lista över komponenter. Returnerar ett spel.

append_component – Tar emot ett spel och en komponent, lägger till komponenten i spelet.

get_game_weight – Tar emot ett spel. Returnerar den sammanlagda vikten av alla komponenter som spelet består av i kg.

print_game – Tar emot ett spel. Skriver ut information om spelet enligt körexemplet nedan, observera att denna funktion anropar print_component för att skriva ut information om de individuella komponenterna.

Din lösning skall följa goda konventioner för implementation och användning av dessa två abstrakta datatyper. Observera att spelet inte känner till den interna representationen av en komponent.

Körexempel:

```
python3 uppgift4.py
Gloomhaven
===========
A fantasy adventure in the town of gloomhaven

Weight: 0.23kg
Map tile 1, weight: 120, description: A tile representing part 1 of the
map
Map tile 2, weight: 110, description: A tile representing part 2 of the
map
Hero piece, weight: 4, description: A miniature representing the player
character

Descent
===========
A fantasy adventure in the land om Tamriell

Weight: 0.11kg
Map tile 2, weight: 110, description: A tile representing part 2 of the
map
```

Uppgift 5

Picard spelar mycket av en förenklad variant av spelet Boggle. Tyvärr vill ingen spela med honom. Han ska därför skapa en datorstyrd motståndare som han kan tävla mot. För att kunna bedöma hur bra hans datorstyrda motståndare är så vill han att du skapar ett program som kan hitta alla möjliga ord på ett Bogglebräde. I denna variant av Boggle kan man hitta ord vertikalt och horisontellt men inte diagonalt.

Ett bräde består av 4 x 4 bokstäver som matas in i början av programmet:

```
python3 uppgift5.py hejstvartarskels
```

Blir ett bräde som ser ut som följande:

```
h e j s
t v a r
t a r s
k e l s
```

I filen ordlista.txt finns en samling av ord från runeberg.org (ca 40000). Ordlistan är strukturerad så att det finns ett ord följt av ett blanksteg följt av ett heltal. Du kan ignorera heltalet, det är inte relevant för uppgiften.

Körexempel:

```
python3 uppgift5.py hejstvartarskels
tars
va
jarl
```