CSC, KTH
DD1314 våren 2017 (Python)

101 Belysning av klot

P-uppgiften ska göras individuellt. Läs CSC:s hederskodex innan du börjar!

Varudeklaration: Skärmhantering, simulering av en matematisk modell. OBS! Kräver en del matematikkunskaper.

Ett klot med centrum i origo och med radien r är belyst av ljus som faller in parallellt med den linje som går igenom punkten (x_0, y_0, z_0) på klotets yta och origo. Uppgiften är att skriva ett program som beräknar belysningsvariationen på klotet och som ger en snygg bild av det belysta klotet på datorn.
Belysningsintensiteten b i en punkt (x, y, z) på klotet är proportionell mot $\cos(v)$ där v är ljusets infallsvinkel i punkten. Låt proportionalitetskonstanten vara 1, vilket innebär att belysningen b kommer att anta värden mellan -1 och 1.
Vid utskrift av belysningen används olika intensitet för att markera olika ljusintensitet. Exempel på en teckenfördelning du kan använda dig av (den skall naturligtvis vara enkel att ändra i ditt program):

`'M'`	markerar mörker, dvs obelyst punkt där			b	$\leq$	0
`'*'`	markerar en ganska mörk punkt	0	$<$	b	$\leq$	0.3
`'+'`	markerar en något ljusare punkt	0.3	$<$	b	$\leq$	0.5
`'-'`	markerar en ganska ljus punkt	0.5	$<$	b	$\leq$	0.7
`'.'`	markerar en ljus punkt	0.7	$<$	b	$\leq$	0.9
`' '`	markerar en mycket ljus punkt	0.9	$<$	b	$\leq$	1

Belysningen $b = \cos(v)$ i punkten (x, y, z) beräknas med skalärprodukten

$$b = (x \cdot x_0 + y \cdot y_0 + z \cdot z_0)/r^2$$

Skriv ett program som:

- Läser in värden på r (klotets radie), x_0, y_0 (definierar ljuskällan). I grunduppgiften kan dessa värden läsas in i terminalfönstret.
- Därefter beräknas z_0 enligt formeln.

$$z_0 = \sqrt{r^2 - x_0^2 - y_0^2}$$

- Därefter ska programmet genomlöpa alla x- och y-värden i intervallet $(-r, r)$ och för varje talpar (x, y) beräkna motsvarande positiva z-koordinat på klotytan. Tänk på att även här kontrollera rotuttrycket. Blir resultatet negativt under rotuttrycket, ligger punkten utanför klotets yta (låt $b = 0$ där).

$$z = \sqrt{r^2 - x^2 - y^2}$$

  Dela upp intervallen i ett lagom stort antal steg (t.ex. 70). Det ska vara lätt att ändra antalet steg i både x- och y-led. Använd konstanter för detta.

VGV

Extrauppgift, betyg C: Kontrollerar indata, uttrycket nedan under rottecknet ska ej bli negativt. Om uttrycket under rottecknet blir negativt ligger ljuskällan felaktigt och nya indata för x_0 och y_0 måste inhämtas.

Extrauppgift, betyg B: Hur ser klotets skugga ut? Lägg till skuggan i bilden! Använd ett annat tecken än de du använt i klotbilden, så att det går att skilja skuggan och klotet åt.

Extrauppgift, betyg A: Rita klotet i ett grafikfönster. Låt användaren flytta ljuskällan genom att klicka med musen på klotet. Ljuskällan ska bara flyttas om klickningen sker på klotet (inte om man pekar på bakgrunden), annars kommer det inte att gå att beräkna z_0.

CSC, KTH
DD1314 våren 2017 (Python)

115 Kösimulering

P-uppgiften ska göras individuellt. Läs CSC:s hederskodex innan du börjar!

Varudeklaration: Datastrukturer. Händelsestyrd simulering.

Det gäller att simulera kösituationer på det lilla postkontoret i Skruttemåla. Posten öppnar kl 9.00 och stänger kl 18.00. Kunder anländer i genomsnitt var femte minut, dvs sannolikheten för att en ny kund ska komma under en viss minut är 20%. För postexpediten, fru Franco, tar det exakt två minuter att betjäna ett postärende. Hälften av alla kunder har ett enda ärende, en fjärdedel har två ärenden, en åttondel tre ärenden osv.
Vid stängningsdags låser fru Franco dörren men betjänar plikttroget de kunder som står i kö. Därefter för hon dagens kundstatistik (totala antalet kunder, alla kunders sammanlagda väntetid och genomsnittliga väntetiden per kund). Allt detta ska simuleras av ditt program enligt följande exempel:

```
Kl 9.03 kommer kund 1 in och blir genast betjänad
Kl 9.05 kommer kund 2 in och ställer sej i kön som nr 2
Kl 9.07 går kund 1 och kund 2 blir betjänad
Kl 9.09 kommer kund 3 in och ställer sej i kön som nr 2
Kl 9.09 går kund 2 och kund 3 blir betjänad
Kl 9.12 kommer kund 4 in och blir genast betjänad
Kl 9.13 kommer kund 5 in och ställer sej i kön som nr 2
Kl 9.15 kommer kund 6 in och ställer sej i kön som nr 3

                                    ⋮

Kl 18.00 stängs dörren
Kl 18.04 går kund 110 och kund 111 blir betjänad
Kl 18.06 går kund 111
STATISTIK: 111 kunder, kundväntetid 58 minuter = 31 s/kund
```

Alla kunder förutsätts anlända vid hela minuttider, ingen får komma indrällande några sekunder för tidigt eller för sent!

- Vid ankomsten slumpas kundens antal ärenden fram enligt sannolikheten som gavs ovan. Tricket ligger i att köra en loop som bryts med 50% sannolikhet. Varje nytt ärende betyder ökad betjäningstid. Utträdestiden beräknas dock inte förrän kunden ska betjänas.
- Är kön tom när en ny kund anländer, betjänas denna direkt (utträdestiden beräknas).
- Är kön inte tom, ställs kunden i kön.

Programmet ska även kontrollera om den första kundens utträdestid=nutid. Är så fallet är alltså denna kund färdigbetjänad och nästa kund i kön kan betjänas.

För att programmet ska bli mer dynamiskt så ska alla parametrar som finns i programmet (öppet tider, minuter per kund, sannolikhet att en ny kund kommer, osv) ligga i en fil.

Extrauppgift, betyg C: Inför möjligheten att ändra på av problemets parametrar (öppettider, ankomstsannolikhet), med hjälp av inmatning. Inför även felkontroll av dessa parametrar.

Extrauppgift, betyg B: Ibland - ganska sällan - blir fru Francos postkontor rånat. Var 1000:e (i genomsnitt) person som kommer in på posten har ondskefulla tankar. Desperadorånaren rusar in och skjuter vilt omkring sig. Kön skingras åt alla håll. En del blir nedskjutna. Alla försvinner således ur kön. Fru Franco, som har svart bälte i karate, försöker givetvis övermanna rånaren. Oftast lyckas hon. Då får postkontoret en PR-kick, och den närmaste tiden kommer fler kunder. Sannolikheten att en kund ska komma en viss minut direkt efter rånet blir 50%, och avtar sedan successivt ner mot de vanliga 20%. Om hon inte lyckas, kommer färre kunder den närmaste tiden. Sannolikheten att en kund kommer en viss minut startar från 5% och går successivt upp mot 20%. Hur många som blir nedskjutna, och huruvida fru Franco lyckas övermanna rånaren slumpas på lämpligt sätt.

Extrauppgift, betyg A: Gör ett grafiskt användargränssnitt där man kan välja parametrar och klicka sig fram i simuleringen.

Datafiler och hjälpfiler: `www.csc.kth.se/~lk/P/random.txt`

CSC, KTH
DD1314 våren 2017 (python)

122 Livet hos matematiska celler

P-uppgiften ska göras individuellt. Läs CSC:s hederskodex innan du börjar!

Varudeklaration: Datastrukturer, filhantering.

En familj organismer består av celler (rutor) i ett rutnät som kan beskrivas av en matris. Cellerna föds och dör enligt bestämda levnadsregler och vi vill undersöka hur familjens fortbestånd förändras under ett antal generationer. Varje cell har åtta grannar:

```
1 2 3
4 * 5
6 7 8
```

Överlevnad

- Varje cell med två eller tre levande grannar kommer att överleva till nästa generation.
- En cell med mer än tre grannar dör av överbefolkning.
- En cell med mindre än två grannar dör av ensamhet.

Födelse

- En tom cell som har exakt tre grannar kommer att födas och bli en levande cell i nästa generation.

Skriv ett program som först läser in antalet önskade generationer samt storleken på matrisen. För att slippa tråkiga och tidsödande inmatningar ska du lägga upp filer med cellernas koordinater. En fil kan se ut så här:

```
Format: x-koord y-koord
============================================================
2 3
2 4
2 5
 ⋮
```

Presentationen kan enklast ske via ascii-grafik, dvs vanliga tecken och blanka skrivs ut så att de tillsammans ser ut som en enkel figur.

Programmet skall vid en sådan presentation rita ut matrisen och dess levande innevånare på ett illustrativt sätt. Det är lämpligt att lägga in en pausfunktion så att användaren hinner se alla utritningar, t.ex. att användaren trycker retur innan nästa bild ritas upp.

Tips: Antag att du har valt att använda dig av en 15x15-matris. För att undvika vissa konstigheter ute i kanterna kan du då jobba med en 17x17-matris, detta för att kunna införa fiktiva celler i 0:te och 16:e position (så att alla verkliga celler får 8 grannar).

Exempel på starttillstånd som ger en trevlig fortsättning:

```
18 levande celler:
    ----------
    ----------
    ---*--*---
    ---****---
    --*----*--
    --*-**-*--
    --*----*--
    ---****---
    ----------
    ----------
```

```
CHESHIREKATT. Utskrift varje generation.
Lägg katten mitt i rutnätet. Efter några
generationer är bara leendet kvar.
```

```
5 levande celler:
   -*--------
   --*-------
   ***-------
   ----------
   ----------
    GLIDARE.
Vid utskrift av var 4:e generation
glider denna figur snett nedåt.
```

```
9 levande celler:
   ----*-----
   -----*----
   *----*----
   -*****----
   ----------
    RYMDSKEPP.
Flyttar sig horisontellt om
utskrift görs var 4:e generation.
```

Extrauppgift, betyg C: Programmet ska kontrollera att koordinaterna ligger inom tillåtet intervall. Användaren skall även kunna bestämma hur ofta han/hon vill se processens fortskridande. Användaren matar in ett heltal, ex. 4 , vilket tolkas som att användaren vill se var 4:e generation.

Extrauppgift, betyg B: Inför följande förbättringar av ditt program

- I stora tomma områden är det onödigt att kontrollera varje cell. Hitta på ett sätt att hålla reda på vilka delar som är just nu inaktiva, och skippa dessa vid genomgången.
- Istället för att ha döda kanter kan man tänka sig att matrisen är klistrad på en cylinder så att vänster kant sitter ihop med höger (och övre kanten med den undre). Inför detta och hitta ett starttillstånd som demonstrerar övergången.

Extrauppgift, betyg A: Gör ett grafiskt gränssnitt, där förloppet ritas ut i ett rutnät istället. Lägg också till två knappar och ett textfält; En av knapparna ska stega bilden till nästan utritning, den andra ska rensa matrisen och i textfältet ska användaren mata in hur många generationer som ska gå mellan utritningarna. Gör också så att man kan skapa och/eller döda celler genom att trycka på rutorna.

Datafiler och hjälpfiler: `www.csc.kth.se/~lk/P/cheshirekatt.txt`
`www.csc.kth.se/~lk/P/glidare.txt`
`www.csc.kth.se/~lk/P/rymdskepp.txt`

CSC, KTH
DD1314 våren 2017 (Python)

127 Platsbokning på SJ

P-uppgiften ska göras individuellt. Läs CSC:s hederskodex innan du börjar!

Varudeklaration: Filhantering, datastrukturer.

Skriv ett program som hjälper SJ med platsbokning i en järnvägsvagn. På skärmen ska en bild, liknande den nedanstående, ritas upp. Observera den fiffiga numreringen som gör att platsnummer i följd ger intilliggande platser. Modulo kan vara till hjälp för att få till kolumner och rader. Välj själv om vagnen skrivas ut antingen på bredden eller på höjden. Redan bokade platsnummer markeras på lämpligt sätt på skärmen, t ex genom att numret omges med två stycken '*'.

1	8	9	16	17	24	25	32
2	7	10	15	18	23	26	31
TYST AVD							
3	6	11	14	19	22	27	30
4	5	12	13	20	21	28	29

Med hjälp av följande meny ska användaren kunna boka respektive avboka platser samt skriva ut biljetter på de senaste bokningarna (dvs de som inte redan skrivits ut).

```
Vad vill du göra?

    • Boka, skriv 'B', på samma rad följt av önskat antal biljetter.
    • Avboka, skriv 'A', på samma rad följt av ett platsnummer.
    • Skriva ut de senast bokade biljetterna, skriv 'S'.
    • Avsluta, skriv 'Q'

Ditt val:
```

Efter varje bokning / avbokning ska bokningsläget uppdateras på skärmen.

Detta program behöver bara skriva ut biljetter på en enda sträcka, t. ex. mellan Stockholm och Göteborg. Platsbiljetterna skrivs ut på en egen "biljettfil" med exempelvis följande utseende:

```
PLATSBILJETT
 Sth-Gbg 9.05
   Plats 19
TYST AVDELNING
   Mittgång

VGV
```

Extrauppgift, betyg C: Felmeddelande ska skrivas ut vid felaktiga inmatningar t ex (fler finns):

- Bokning av fler platser än vad som finns kvar.
- Avbokning av en obokad plats.
- Avbokning av platsnummer som ligger utanför intervallet $1 \ldots 32$
- "Utskrift" väljs, trots att inga bokningar har gjorts.

Extrauppgift, betyg B: Inför fler sträckor och avgångstider. Ett tåg består rimligtvis av flera vagnar, så se till att ditt program bokar platser på tåg. Ett tåg har ett unikt tågnummer, som kan användas för att hålla ordning på textfilerna.

Se till att programmet försöker ordna så att bokade platser hamnar intill varann. Om det inte går ska programmet fråga om det är OK med spridda platser.

Bokningsläget (som består av de bokade platsernas nummer) för varje sträcka, tåg och avgång ska sparas på fil.

Aktuell fil läses in igen vid nästa bokning så man inte riskerar dubbelbokningar.

Extrauppgift, betyg A: Gör ett grafiskt användargränssnitt. Alla bokningar ska ske via musklickningar. Lägg också till en knapp för att skriva ut bokade biljetter.

CSC, KTH
DD1314 våren 2017 (python)

134 Springarens vandring på schackbrädet

P-uppgiften ska göras individuellt. Läs CSC:s hederskodex innan du börjar!

Varudeklaration: Datastrukturer, randvillkor.

Skriv ett program som följer en springares vandring över ett 64 rutors schackbräde! Springaren följer schackreglerna, dvs den förflyttar sig antingen två steg horisontellt och ett steg vertikalt eller två steg vertikalt och ett steg horisontellt. Under hela vandringen får springaren inte komma till en plats mer än en gång.

Användaren får ange vilken ruta springaren ska starta på. Det finns högst 8 stycken platser som springaren kan gå till från varje ruta. Du skriver lämpligen en klass vars fält representerar schackbrädet och vars metoder tex. förflyttar springaren. Står vi på en kantruta kan vi inte flytta springaren i vissa riktningar, ty då hamnar vi utanför brädet. I tipset finner du en variant att lösa detta problem. Programmet ska plocka ut de godkända alternativen och sedan med hjälp av slumpen välja vilken ruta som springaren ska gå till.

Exempel:

```
nyPlats = random.randrange(antalOKplatser)
```

Du får numrera de godkända platserna enligt en fast ordning, tex medsols. Programmet tar slut då springaren inte har någon plats att flytta till. Användaren ska starta programmet genom att ange en startpunkt på schackbrädet. Därefter ska brädet skrivas ut. I presentationen av schackbrädet ska det tydligt framgå:

- Rutnätet, varje koordinat har en egen ruta.
- Springarens förflyttningar ska vara markerade, en ruta kan då innehålla:

  Tomrum , springaren har inte besökt denna ruta.

  Ett tal , springaren har besökt denna ruta, i steg nr ...

- Koordinater i kanterna.

Tips: Schackbrädet kan lämpligen representeras av en matris som förutom det verkliga schackbrädet (rad 2 – 9, kolumn 2 – 9) har ytterligare två rader och kolumner runt omkring (som flyttbuffert). Matrisen får då indexgränserna (0..11). På raderna 0 och 1 såväl som i kolumnerna 10 och 11 sätts alla element till -1 för att markera att rutorna ligger utanför brädet. För att markera rutornas tillstånd ska du använda dig av någon konvention.

T ex kan olika tal ge information om rutan:

-1 Rutan ligger utanför schackbrädet.

0 Rutan ligger på schackbrädet, springaren har aldrig varit här.

k Rutan ligger på schackbrädet, springaren har varit här tidigare, i steg nummer k (där k är ett tal > 0).

Utskriften från programmet kan se ut så här:

```
Springarens vandring:
   ----------------------------------------
   |   --------------------------------   |
   |8  |   |   |   | 2 |   |   |   |   |   |
   |   --------------------------------   |
   |7  |   |   |   |   |   | 3 |   |   |   |
   |   --------------------------------   |
   |6  |   |   | 1 |   |   |   |   |   |   |
   |   --------------------------------   |
   |5  |   |   |   |   |   |   |   |   |   |
   |   --------------------------------   |
   |4  |   |   |   |   |   |   |   |   |   |
   |   --------------------------------   |
   |3  |   |   |   |   |   |   |   |   |   |
   |   --------------------------------   |
   |2  |   |   |   |   |   |   |   |   |   |
   |   --------------------------------   |
   |1  |   |   |   |   |   |   |   |   |   |
   |   --------------------------------   |
   |     A   B   C   D   E   F   G   H     |
   ----------------------------------------
```

Extrauppgift, betyg C: Ge användaren möjlighet att mata in en egen springarvandring. Kontrollera att inmatade koordinater motsvarar en tillåten ruta.

Extrauppgift, betyg B: Programmet ska uppdatera en ”high-score-fil” mellan körningarna. Textfilen läses in vid programstart. Om körningen resulterade i fler steg än det hittills största, som är lagrat i textfilen, ska det nya antalet steg lagras på textfilen.

Extrauppgift, betyg A: Ge programmet en snyggare presentation genom att visa resultatet i ett separat grafikfönster. Låt användaren få se ett steg i taget.

CSC, KTH
DD1314 våren 2017 (python)

133 Sänka skepp

P-uppgiften ska göras individuellt. Läs CSC:s hederskodex innan du börjar!

Varudeklaration: Datastrukturer. Viss skärmhantering. I extrauppgiften tillkommer sortering.

Du ska skriva ett program som slumpvis lägger ut en mängd fartyg på en yta. Användaren ska sedan kunna beskjuta ytan och programmet ska meddela resultatet (envägs beskjutning). Spelet avslutas då användaren så önskar eller efter att alla fartyg är sänkta. Regler för utläggning av fartyg:

- Fartygen måste ligga helt inom spelplanen.

Programmet skall efter varje skott:

- Skriva ut resultatet av skottet, träff eller bom.

Användaren ska via menyer kunna göra sin olika val. Huvudmenyn kan se ut så här:

```
Dina valmöjligheter (1-3):
1) Beskjuta fiendefartyg
2) Fuska lite, tjuvkika på fiendefartygen
3) Avsluta
Ditt val (1-3):
```

De olika menyerna innebär:

Beskjutning mot fiendefartygen: På skärmen ska spelplanen och skottmarkeringarna visas i en schackbrädesliknande figur. Användaren ska få skjuta ett eller flera skott och därefter återgå till huvudmenyn. Under spelplanen ska träffprocenten skrivas ut.

Fuska lite, tjuvkika på fiendefartygen: Alla spelare har olika moral och alla måste tillfredsställas. Spelplanen presenteras med fiendefartygen fullt tydligt markerade tillsammans med skottmarkeringarna.

Avsluta: Användaren ska få möjlighet att titta på de ej sänkta fartygens lägen före avslutning. Spelplanen presenteras på samma sätt som vid tjuvkiken.

Du bestämmer, som programkonstruktör, enväldigt spelplanens storlek (t ex 8x8). Likaså bestämmer du fartygens antal samt storlek, vilket presenteras i början av programmet (det ska finnas fartyg av olika längder, t.ex. 1 - 5 rutor). Du skall programmera på ett sådant sätt att man i efterhand lätt kan förändra fartygsuppsättningen. I presentationen av spelplanen ska det tydligt framgå:

- Rutnätet, varje koordinat har en egen ruta, se bilden nedan.
- Resultaten av skotten, t.ex.

  Träffar markeras med '#'

  Bommar med 'o'

  Fartyg , (ej träffade) med 'X'

- Koordinater i kanterna.

Tips: För att representera rutornas tillstånd använder vi oss av någon lämplig konvention. T ex kan heltal ge information om rutan:

0 Rutan ej beskjuten, inget fartyg ligger här.

1 Rutan ej beskjuten, del av ett fartyg ligger här.

2 Rutan beskjuten, bom.

3 Rutan beskjuten, träff.

Det kan vara praktiskt att också lagra mer information för varje ruta. Om det i rutan ligger en del av ett fartyg kan man t ex vilja veta hur långt fartyget är och på vilken ledd det ligger (horisontellt eller vertikalt).

I filen `www.csc.kth.se/~lk/P/random.txt` får du tips till slumpfunktionen, som du använder till att slumpa fram fartygens positioner. För att på ett enkelt sätt kunna presentera och ändra fartygens storlek kan du skriva ihop en textfil med fartygens längder. Den kan se ut så här:

```
% Format: heltal som anger fartygens längd (antal rutor)
5 3 2 1 1
```

Presentationen ska vara tydlig.

Extrauppgift, betyg C: Inför felkontroll av menyalternativ och inmatning.

Skriva ut ett felmeddelande om skottet var ogiltigt, dvs utanför planen eller rutan redan beskjuten. Användaren får i detta fall skjuta på nytt.

Extrauppgift, betyg B: Uppdaterade regler för spelet:

- Två fartyg får aldrig ligga precis intill varandra, utan det måste alltid finnas tomma rutor emellan.
- Om det blev en träff och fartygets samtliga rutor är träffade är fartyget sänkt. Detta ska meddelas användaren. Dessutom ska programmet markera "beskjuten" på alla kringliggande rutor eftersom det inte kan ligga några fartyg där.

Extrauppgift, betyg A: Gör ett grafiskt gränssnitt där man klickar för att beskjuta.

Datafiler och hjälpfiler: `www.csc.kth.se/~lk/P/random.txt`

CSC, KTH
DD1314 våren 2017 (Python)

139 Tidtabeller för en pendeltågslinje

P-uppgiften ska göras individuellt. Läs CSC:s hederskodex innan du börjar!

Varudeklaration: Filhantering, listor.

På en pendeltågslinje sätter man då och då in nya lok med allt starkare motorer. Det betyder att tidtabellen måste förnyas varje gång. Din uppgift blir att skriva ett program som tillverkar nya tidtabeller för linjen. Tidtabellen ska innehålla stationsnamn och avgångstider för varje station längs linjen, t ex Floda 09.54.
Indata är stationsnamnen, avstånden från startstationen till de olika stationerna på linjen samt data på tågets prestanda. För att få med av- och påstigningstiden ska du använda följande modell, som ger väntetider på 0–1 minuter:

avgångstiden = heltalsdelen av ankomsttiden + 1 (minuter)

Lägg upp en fil med alla indatavärden och ändra i denna då tågprestanda eller linjesträckningen ska förnyas. Exempel på hur filen kan se ut:

```
Prestanda, format:
acc (0.4 - 0.8 m/s2) / retardation (1.2 - 2.0 m/s2)
maxhast (30 - 45 m/s)
============================================================
0.5 1.5 35
============================================================
Tabellformat:
antal stationer
stationsnamn (max 15 tkn) / avstånd från startstationen (km):
============================================================
 11
 Arby     0
 Bedinge  3.57
 Cebro    6.38
 Degum    8.67
 Ekö      12.80
 Floda    18.24
 Guldö    24.59
 Håsta    29.42
 Iby      30.73
 Jituna   34.06
 Kåvik    36.95
```

Alla dagar

Endast ett tåg är i trafik. Första avgången mot Kåvik är kl 08.35 från Arby. När tåget anlänt till Kåvik, väntar det på stationen i 10 minuter innan det vänder tillbaka mot Arby. Vid ändstationerna beräknas alltid avgångstiden ligga 10 minuter efter ankomsttiden. Sista tåget från Kåvik ska ha en avgångstid före midnatt. Efter ankomst till Arby står det kvar där över natten. Detta innebär att tåget kommer att åka fram och tillbaka ganska många gånger.

Naturligtvis ska det vara någorlunda lätt att ändra dessa data. Man skulle kunna starta tågen senare, låta tåget vänta längre på station osv.

Tågets prestanda anges med följande parametrar:

a	tågets acceleration (m/s^2)
r	tågets retardation (m/s^2)
v_{max}	tågets maxhastighet (m/s)
s	sträckan mellan två stationer (m)
s_0	sträckan som tåget behöver för att nå sin maxhastighet $\frac{v_{max}^2}{2 \cdot a}$ (m)
s_1	sträckan som tåget behöver för att bromsa till stillastående vid maxhastighet $\frac{v_{max}^2}{2 \cdot r}$ (m)

För restiderna mellan två stationer gäller då:

$$t = \begin{cases} \sqrt{2 \cdot s \left(\frac{1}{a} + \frac{1}{r}\right)} & \text{för } s \leq s_0. \\ v_{max} \cdot \left(\frac{1}{a} + \frac{1}{r}\right) + \frac{s - s_0 - s_1}{v_{max}} & \text{för } s > s_0. \end{cases}$$

Tips: Beräkna tidsintervallen i minuter mellan stationerna, med hänsyn tagen till av- och påstigningstiderna (se tidigare givna modellen) och lägg upp dem i en lista.

Gör en funktion som beräknar n stycken klockslag bestämda av startklockslaget och tidsintervallen. Lägg upp dem i en lista. Använd strukturen ovan för att beräkna tiden δ som det tar för ett tåg att gå hela linjen fram & tillbaka och vara startklar på nytt. Avgångstiderna för hela dagen från en viss station kan sedan lätt beräknas med hjälp av δ och morgonturens tider.

Lägg upp varje stations avgångstider i en lista. I utskriften ska stationens namn stå längst till vänster med ca 10 klockslag per rad. Tänk på att antalet avgångar kan vara större än vad som får plats på en rad. Utskriften får i detta fall delas upp på flera sidor. Utskriftsformatet för tiderna: 07.52, 08.04; tänk på nollorna!

Var noga med att kolla på B-uppgiften innan du designar dina klasser, det är viktigt att du inte har en struktur som inte passar med de högre kraven.

Extrauppgift, betyg C: Låt användaren kunna ange nya värden för tåget än de som finns på fil. Inför felkontroll av indata.

Extrauppgift, betyg B: För att göra uppgiften intressantare så ska du nu skilja på vardagar och helgdagar, den tidigare beskrivningen gäller nu för helgdagar och följande gäller för vardagar.

Vardagar

Nu finns det tre tåg i trafik (tre ggr tätare trafik). Första avgång från Arby är kl 05.07. De övriga avgångarna ska ligga så att jämna intervall mellan tågen erhålls. Alla tåg utgår från Arby, dvs morgontåget från Kåvik kan inte avgå förrän 10 minuter efter att det första tåget från Arby kommit dit. I övrigt gäller samma kriterier som för helgdagarna.

Extrauppgift, betyg A: Gör ett grafiskt användargränssnitt till programmet!

CSC, KTH
DD1314 våren 2017 (Python)

140 Sålda biobiljetter

P-uppgiften ska göras individuellt. Läs CSC:s hederskodex innan du börjar!

Varudeklaration: Filhantering, datastrukturer, sortering.

För att visa din kulturella sida har du investerat några av dina miljoner i diverse biografer. Varje dag får du veta hur många biljetter som sålts till föreställningen kvällen innan och du vill skriva ett program som bearbetar och presenterar denna information.

Skriv först ihop en fil med följande information för varje biograf:
biografens namn, totalt antal platser i salongen, biljettpriser för vuxna, pensionärer och barn. Exempel:

```
Draken
1038
80/65/50
```

Biljettpriserna varierar från biograf till biograf. Vuxenbiljetten är alltid dyrast och barnbiljetten billigast och priserna är alltid i hela kronor.

Ditt program ska läsa in informationen från filen.

Sen ska användaren få ange antal sålda biljetter i varje prisklass för varje biograf. Så här kan det se ut när man kör programmet:

```
 1. Draken
 2. Roy
 3. Sture
 4. Zita
 5. Kino
Välj biograf: 3
Sålda vuxenbiljetter på Sture: 83
Sålda pensionärsbiljetter på Sture: 128
Sålda barnbiljetter på Sture: 4
 1. Draken
 2. Roy
 3. Zita
 4. Kino
Välj biograf:
```

När användaren har matat in data för alla biografer ska programmet för varje biograf skriva ut summan av biljettintäkterna och beläggning (i procent). Biograferna ska sorteras i fallande ordning efter beläggning (den som har utsålt hamnar alltså först).
Dessutom ska summan av alla biografers biljettintäkter skrivas ut.

Tänk på flexibiliteten - filen kan när som helst uppdateras med höjda biljettpriser och fler biografer så ditt program måste kunna hantera detta.

Extrauppgift, betyg C: Inför felkontroll av användarens inmatning. Kontrollera också att filen existerar och att filens data är rimliga, dvs att varje biograf har namn, antal platser och tre olika biljettpriser, och att biljettpriserna följer reglerna ovan (barnbiljetter billigast osv).

Extrauppgift, betyg B: Biljettförsäljaren räknar ihop kassan efter föreställningen och vill ur kassans storlek avgöra hur många biljetter av varje sort som har sålts. Det kan finnas många lösningar till detta problem och ibland ingen alls (på grund av felräkning i kassan).

Uppgiften blir att skriva metoder som givet de tre biljettpriserna ovan samt kassans storlek skriver ut alla lösningar. Vi söker heltalslösningar så att:
$antvuxna \cdot vuxpris + antpens \cdot penspris + antbarn \cdot barnpris = kassan$

I programmet ska man sedan i en meny kunna välja om man vill ha statistik för alla biograferna (enligt grunduppgiften) eller räkna ut hur många biljetter som sålts av varje sort för en viss biograf.

Tips: Tar det lång tid att göra beräkningarna? Fundera över vilka permutationer som är rimliga att testa.

Extrauppgift, betyg A: Gör ett grafiskt användargränssnitt till programmet.

CSC, KTH
DD1314 våren 2017 (Python)

145 Varuprisdatabas

P-uppgiften ska göras individuellt. Läs CSC:s hederskodex innan du börjar!

Varudeklaration: Filhantering samt datastrukturer.

I många butiker är varorna inte prismärkta utan bär i stället ett streckkodat varunummer som kassaexpediten läser av med ljuspenna. I butiksdatorns databas finns varans data lagrade på en textfil som kan se ut så här:

```
% Format:
% kod
% namn
% pris antal
===========================================
100
CHIPS
14.90 353
135
STÖVLAR
159 234
 ⋮
```

Expediten läser av varorna med sin ljuspenna, för att få ett kvitto avslutar denne med att mata in ett #-tecken. Så här kan kvittot se ut:

Varunamn	Antal	A-pris	Summa
CHIPS	1	14.90	14.90
VOLVO	2	107000.00	214000.00
STÖVLAR	1	159.00	159.90
CHIPS	1	14.90	14.90
Total	5		214189.70

Ditt program ska uppföra sig på motsvarande sätt. Databasen ska ligga i en varufil, som du skriver ihop själv. Då vi saknar ljuspenna matar vi in varorna via kassaapparatens tangentbord. Inmatningen av det ovanstående inköpet kan då se ut så här:

```
100
280 2
135
100
#
```

För att hålla antalet filläsningar nere ska du läsa in varufilen i en datastruktur. Detta betyder att du uppdaterar datastrukturen kontinuerligt och varufilen endast efter avslutad körning.

Extrauppgift, betyg C: Tänk på att om det finns 20 påsar chips i lager, ska det inte vara OK att mata in först 14 påsar, och sedan 10 till!

Allt eftersom man slår in varunummer kollar programmet att koden finns i databasen samt att det finns tillräckligt antal varor i lager, annars kommer en felutskrift, följt av möjligheten att mata in på nytt.

Kontrollera också att inmatningens syntax är korrekt så att inga tokigheter händer om man råkar tryck på fel knapp.

Extrauppgift, betyg B: Se till att alla varor av samma slag hamnar på samma plats på kvittot! Det innebär att ovanstående inköp skulle ge följande kvitto i stället:

Varunamn	Antal	A-pris	Summa
CHIPS	2	14.90	14.90
VOLVO	2	107000.00	214000.00
STÖVLAR	1	159.00	159.90
Total	5		214189.70

Det händer att kassaexpediten gör felslag. Modifiera ditt program så att det går att ångra inmatade inköp innan kvittot skrivs ut. Man ska givetvis inte behöva ångra allt man matat in efter det felaktiga först. Det ska också vara möjligt att ångra bara en del av sitt inköp, t ex ångra 3 påsar chips, när man köpt 5.

Extrauppgift, betyg A: Gör ett grafiskt användargränssnitt genom vilket all in- och utmatning sker.

CSC, KTH
DD1314 våren 2017 (Python)

154 Wumpus

P-uppgiften ska göras individuellt. Läs CSC:s hederskodex innan du börjar!

Varudeklaration: Datastrukturer, slumptal.

Wumpus är ett enkelt äventyrsspel. Så här ser det ut när man spelar:

```
Du befinner dig i kulvertarna under CSC, där den glupske Wumpus bor.
För att undvika att bli uppäten måste du skjuta Wumpus med din pil
och båge. Kulvertarna har 20 rum som är förenade med smala gångar.
Du kan röra dig åt norr, öster, söder eller väster från ett rum till
ett annat.
Här finns dock faror som lurar. I vissa rum finns bottenlösa hål. Kliver
du ner i ett sådant dör du omedelbart. I andra rum finns fladdermöss
som lyfter upp dig, flyger en bit och släpper dig i ett godtyckligt
rum. I ett av rummen finns Wumpus, och om du vågar dig in i det rummet
blir du genast uppäten. Som tur är kan du från rummen bredvid känna
vinddraget från ett avgrundshål eller lukten av Wumpus. Du får också
i varje rum reda på vilka rum som ligger intill.
För att vinna spelet måste du skjuta Wumpus. När du skjuter iväg en
pil förflyttar den sig genom tre rum - du kan styra vilken riktning
pilen ska välja i varje rum. Glöm inte bort att tunnlarna vindlar sig
på oväntade sätt. Du kan råka skjuta dig själv...
Du har fem pilar. Lycka till!

Jag hör fladdermöss!
Härifrån kan man komma till följande rum: 6 3 2 14
Vill du förflytta dig eller skjuta (F/S)? F
Vilken riktning (N, S, V, Ö)? N
Du känner fladdermusvingar mot kinden och lyfts uppåt
Efter en kort flygtur släpper fladdermössen ner dig i rum 19

Jag känner lukten av Wumpus!
Jag känner vinddrag!
Härifrån kan man komma till följande rum: 2 18 5 15
Vill du förflytta dig eller skjuta (F/S)? S
Pilen lämnar första rummet. Vilken riktning (N, S, V, Ö)? N
Pilen lämnar andra rummet. Vilken riktning (N, S, V, Ö)? Ö
Pilen lämnar tredje rummet. Vilken riktning (N, S, V, Ö)? N
Vill du förflytta dig eller skjuta (F/S)? F
Vilken riktning? S
Du klev just ner i ett bottenlöst hål.
```

Spelaren (den som kör programmet) går runt i kulvertarna, letar rätt på och försöker skjuta Wumpus. Spelet avslutas när spelaren eller Wumpus dör.

Låt varje rum representeras av ett objekt av klassen Rum. I objektet lagras information om vad som finns i rummet (Wumpus, fladdermöss, avgrundshål eller ingenting) och vart gångarna i rummet leder (norra gången leder till rum 17, östra till rum 2 osv). Bilda en lista av rumsobjekt för att representera kulvertarna (varje rum har då ett nummer). Slump får du med random.

I programmets början ska rummens innehåll slumpas fram. Ungefär 20 % av rummen ska innehålla avgrundshål, 30 % ska vara bebodda av fladdermöss och i ett av rummen finns Wumpus. Ett rum kan aldrig innehålla flera faror (Wumpus äter fladdermöss och varken Wumpus eller fladdermössen gillar draget från avgrundshål).

Även kulvertarna ska sättas samman med slumpens hjälp i början av programmet. Gör t ex så här:

- Skapa en lista med alla rumsnummer och blanda den så att rummen kommer i slumpmässig ordning, t ex 5 2 12 18 9 1 3 6 14 19 11 7 17 4 20 16 10 15 8 13
- Koppla ihop rummen i öst-västlig riktning så att östra gången från rum 5 leder till rum 2 och västra gången från rum 2 leder tillbaka till rum 5 osv. Sista rummet i listan ska kopplas ihop med första.
- Slumpa en ny lista och gör samma sak i nord-sydlig riktning. Rummet söder om rummet norr om det rum man befinner sig i ska alltså vara det rum man befinner sig i.

Spara antalet drag som en spelare gör och om man vinner så sparas detta på en high-score fil som uppdateras när spelet avslutas.

Extrauppgift, betyg C: Inför felkontroll för all inmatning från användaren.

Extrauppgift, betyg B: Låt användaren välja svårighetsgrad. Spelet kan göras enklare genom att man sänker sannolikheten för faror, och knepigare genom att låta Wumpus röra sig i kulvertarna (läskigast blir det om Wumpus hela tiden kommer närmare).

Extrauppgift, betyg A: Gör ett grafiskt användargränssnitt (GUI). Du behöver inte slumpa rummen som i grunduppgiften.

CSC, KTH
DD1314 våren 2017 (Python)

155 Tittarsiffror

P-uppgiften ska göras individuellt. Läs CSC:s hederskodex innan du börjar!

Varudeklaration: Sökning, sortering, filhantering.

Många är intresserade av TV:s tittarsiffror, däribland programmakare, programchefer och sponsorer. Du ska skriva ett program som presenterar statistik över tittarsiffror på olika sätt. Så här ska det se ut:

```
---------------------- Meny ------------------
   1. Tio-i-topp-lista
   2. Tittarsiffror för ett visst program
   3. Sluta
   Vad vill du göra? 1

---------------------- Tio-i-topp-lista ------------------
   1. Aktuellt 57 st (60%)
   2. Sportnytt 47 st (49%)
   3. Rapport med väder 34 st (36%)
   4. Ängeln och den laglöse 33 st (35%)
   5. Tippen 20 st (21%)
   6. Myggan 18 st (19%)
   7. Skilda världar 18 st (19%)
   8. Grannar 16 st (17%)
   9. Paradise Beach 16 st (17%)
   10. Norrköpings sommarcafe 12 st (13%)
   Statistik har samlats in från 93 TV-apparater.
   ----------------------------------------------
   Vad vill du göra? 2
```

Ditt underlag får du ur följande två filer: Filen `www.csc.kth.se/~lk/P/tittardata.txt` som innehåller data för en dag insamlade från de TV-apparater som är med i undersökningen. När TV:n är påslagen registrerar den klockslag och inställd kanal vid fasta tidpunkter varje dag. Data från olika apparater ligger i samma fil, men skiljs åt av streckade rader.

```
Format: tid/kanal
=================
19.37/2
19.52/2
21.07/1
-------
16.22/4
16.37/4
```

Filen `www.csc.kth.se/~lk/P/program.txt` innehåller en dags TV-program för flera kanaler (åtskilda av streckade rader). Du får utgå från att filen inte innehåller samma program mer än en gång, samt att inget program går över dygnsgränsen. Det är också Fritt fram att ändra innehållet bäst man vill bara filen är minst lika stor.

```
Format: kanal tid program
==============================
Kanal 1
21.30-21.40 Sportnytt
21.40-22.35 UR: Sommarkvällar med Tidernas Europa
------------------------------------------------------------------
Kanal 2
8.30-9.00 Let's Dance
9.00-10.00 Biggest Loser
```

Datafiler och hjälpfiler: `www.csc.kth.se/~lk/P/tittardata.txt`
`www.csc.kth.se/~lk/P/program.txt`

Extrauppgift, betyg C: Inför felhantering för användarens inmatning.

Extrauppgift, betyg B: Man ska nu också kunna se stapeldiagram över viss kanal

```
Vilken kanal vill du se stapeldiagram för? 2
---------------- Stapeldiagram över antal tittare, kanal 2 ------------------
 8.30- 9.00 Let's Dance                  |***************
 9.00-10.00 Biggest Loser                |*****************
17.50-18.20 Klockan Nio: hos stjärnorna  |*
18.20-19.20 Friends                      |*
19.20-19.30 Regionala nyheter            |**
19.30-20.00 Rapport med väder            |*****************************
20.00-21.00 Fresh Prince of Bel-Air      |***********
21.00-22.25 Teenage mutant ninja turtles |**
22.25-23.15 Hawaii five-0                |**
23.15-23.55 Kalla fakta                  |**
(En stjärna motsvarar 0.9 tittare och längsta stapeln 34 tittare.)
```

Extrauppgift, betyg A: Gör ett grafiskt användargränssnitt till programmet. Använd knappar för att ange olika val och presentera statistiken i snygga diagram.

CSC, KTH
DD1314 våren 2017 (Python)

162 Bibliotek

P-uppgiften ska göras individuellt. Läs CSC:s hederskodex innan du börjar!

Varudeklaration: Datastrukturer, sökning, sortering, filhantering.

Skriv ett program för hantering av enklare biblioteksrutiner. Programmet ska kunna söka efter böcker både med avseende på författare och titel. Man ska även kunna låna och lämna tillbaka böcker, lägga in nya böcker i registret och ta bort gamla böcker, samt skriva ut en lista på skärmen över alla böcker. Böckerna ska sorteras med avseende på författare.

Böckerna ska lagras på en textfil som du får skriva in själv. Din fil behöver inte innehålla fler än tio böcker men ditt program ska gå att använda även för ett stort antal böcker.

Programmet ska komma ihåg vilka böcker som är utlånade även om användaren stänger av och startar om programmet på nytt.

Exempel:

```
Välkommen till biblioteksprogrammet!
   T  söka på Titel.
   F  söka på Författare.
   L  Låna bok.
   Å  Återlämna bok.
   N  lägga in Ny bok.
   B  ta Bort bok.
   A  lista Alla böcker.
   S  Sluta.

Vad vill du göra? F
   Vilken författare vill du söka efter? Martin
   Hittade 2 böcker
   Martin: Agile software development (utlånad)
   Martin: Clean Code

Vad vill du göra? L
   Ange titeln på den bok du vill låna: Clean Code

Vad vill du göra? s
   Välkommen åter!
```

Tips: Skriv programmet i etapper. Börja med att läsa in böckerna från fil och skriva ut dem igen. Utöka sedan programmet stegvis.

Extrauppgift, betyg C: Inför felkontroll för användarens inmatning och filers existens.

Extrauppgift, betyg B: Inför en användardatabas där man måste registrera en personlig användare för att få låna böcker. Låt nu programmet hålla reda på vem som lånat boken och vilket datum den ska återlämnas.

En person har namn, personnummer, emailaddress och typ. Två olika typer av användare skall finnas. Först har vi vanliga användare, som ska kunna låna och lämna tillbaka böcker till biblioteket. Sedan har vi typen administratör, som även (likt grunduppgiften) kan lägga till och ta bort böcker ur biblioteket, samt lägga till nya användare, och ta bort användare som inte sköter sig!

En administratör ska kunna be om en lista på alla personer som lånat böcker, och vilka de lånat. Böcker som borde varit tillbakalämnade ska markeras. Man ska också kunna få en lista på enbart personer som har böcker hemma vars datum gått ut tillsammans med dessa böcker och hur mycket personen är skyldig i böter. Kolla upp bötesregler på biblioteket!

Tips: Beroende på vilken användare som loggar in i bibliotekets söktjänst ska alltså olika menyer visas.

Tänk på vad som händer om en administratör tar bort en användare som fortfarande har lånade böcker!

Extrauppgift, betyg A: Gör ett grafiskt användargränssnitt till programmet.

CSC, KTH
DD1314 våren 2017 (Python)

166 Telefonregister

P-uppgiften ska göras individuellt. Läs CSC:s hederskodex innan du börjar!

Varudeklaration: Datastrukturer, sökning, sortering, filhantering.

Ett vanligt problem är att man vill lägga upp ett telefonregister där man snabbt kan leta upp telefonnumret för en viss person eller personen som har ett visst telefonnummer. Man vill också kunna lägga till personer i registret, ta bort personer ur registret samt kunna ändra en persons telefonnummer och adress. Ditt program skall:

1. Läsa in ett register från en fil med namn, telefonnummer och adress för ett antal personer.
2. Så länge användaren vill köra så skall programmet fråga om man vill leta efter ett telefonnummer eller ett namn, lägga till nya uppgifter, ta bort uppgifter, ändra uppgifter (telefonnummer och adress) eller ha en sorterad (namnordning) utskrift av registret.
3. Innan programmet slutar skriva ut uppgifterna på fil igen, inklusive eventuella nya uppgifter och ändringar.

Filen med personuppgifterna skriver du själv in. Du bör ha minst 15 olika personer. Använd en textfil där varje person tar upp 4 rader. De olika raderna innehåller efternamn, förnamn, telefonnummer och adress. Genom att använda en textfil kan du själv lätt skriva in data. Filen behöver inte vara sorterad, men får gärna vara det. Exempel:

```
Andersson
Anders
123456
Testvägen 4, Huddinge
Jansson
Jan
324567
Klutvägen 3, Bromma
...
```

Den sorterade listan skrivs lämpligen med en person på varje rad:

```
Efternamn   Förnamn   Telefon         Adress
=================================================
Andersson   Anders    123456    Testvägen 4, Huddinge
Jansson     Jan       324567    Klutvägen 3, Bromma
...
```

När man ändrar data om en person, vill man inte vara tvungen att skriva in det som är oförändrat (t ex adressen, om man bara ska byta telefonnummer) på nytt, utan bara det som ska ändras. Det är viktigt för sökningen att uppgifter som skrivs in av användaren, både vid nyinmatning, ändring och sökning, omvandlas till ett enhetligt format. Se till att ta bort inledande och avslutande blankslag, och se till att alla telefonnummer får samma form, t ex helt utan andra tecken än siffror!

Sökning efter ett visst telefonnummer går till så att användaren matar in ett telefonnummer, programmet letar upp personen med det telefonnumret och sedan skrivs namn, adress och telefonnummer ut för personen. Sökning efter namn går till på motsvarande sätt.

Ändring av en persons uppgifter går till så att man matar in personens för och efternamn. Motsvarande person söks upp och programmet läser in de nya uppgifterna för telefonnummer och adress. Om du vill kan du göra så att man har ett ändringsalternativ för telefonnummer och ett för adressen.

När man kör programmet skall man kunna få se en hjälptext genom att ange något kommando, t.ex. ett '?'.

Tips: Skriv programmet i etapper. Börja med att läsa in personerna från fil och skriva ut dem igen. Utöka sedan programmet stegvis.

Extrauppgift, betyg C: Inför felkontroll för användarens inmatning och filers existens.

Extrauppgift, betyg B: Man kan ju vilja ha flera telefonregister, t ex ett över badmintonklubben man är ordförande i och ett privat, över vänner. Se till att man kan ha flera register med olika titlar. I början av körningen ska man få välja:

- Använda ett register (som i grunduppgiften). Man kommer vidare till en meny med registernamn. I denna ska man på ett enkelt sätt (inte genom att skriva hela registernamnet) kunna välja ett register eller att gå tillbaka till huvudmenyn. De olika registren lagras i olika filer, lämpligen med namn som 'badminton.reg' och 'vanner.reg' om man har registren 'badminton' och 'vänner'. Byt ut å,ä och ö i filnamnen. Du ska också ha en fil med register över registren, d v s med alla registernamnen. Denna fil läser du in när programmet startar. Om användaren sedan väljer att använda ett register, laddas det valda in.
- Skapa ett nytt register. Se till att ett register med det valda namnet inte finns redan.
- Samköra två register. Här ska menyn med registernamn (se ovan) komma upp. Man ska få välja två register, och sedan komma vidare till menyn:

  Lista på alla som finns i båda (de valda) registren.

  Lista på alla som finns i något av (de valda) registren, utan dubbletter (union).

  Tillbaka till registernamnsmenyn. Du får utgå från att om en person finns med i flera register, så har han samma personuppgifter i alla.

Extrauppgift, betyg A: Gör ett grafiskt användargränssnitt till ditt program.

CSC, KTH
DD1314 våren 2017 (Python)

168 Minröjning

P-uppgiften ska göras individuellt. Läs CSC:s hederskodex innan du börjar!

Varudeklaration: Datastrukturer, rekursion, grafik.

Trots att det är riskfyllt tänker Osquar sommarjobba som minröjare. Tillsammans med sin minhund Trofast går inte Osquar på en endaste mina. Innan Osquar och Trofast kommer till en plats ger nämligen Trofast skall lika många gånger som det finns minor runt platsen. På så sätt kan Osquar bedöma vart han och Trofast skall gå härnäst. Trofast, som är en röjig hund vill gärna känna vittringen av minor. Skulle det vara så att det inte finns någon mina i närheten springer han runt och nosar upp minorna runt omkring. Helt tomma ytor genomsöks automatiskt av Trofast. Osquar får lugnt vänta på sin röjande kamrat.

Ett typiskt minfält kan se ut så här om tio minor placeras ut slumpvis på 64 platser:

```
M = Här ligger det en mina
1 - 8 = Antalet minor som angränsar till denna ruta
tom ruta = ingen mina i närheten
```

	1	2	3	4	5	6	7	8
A	M	M	1					
B	2	2	1			1	1	1
C		1	2	2	1	2	M	2
D	1	2	M	M	2	2	M	2
E	M	2	4	M	3	1	1	1
F	1	1	2	M	2		1	1
G			1	1	1		1	M
H							1	1

Ditt program skall

- Slumpvis placera ut ett antal minor på ett fält. Fältets storlek och antalet minor skall enkelt kunna anges av användaren (Osquar). Fältet ska visas på skärmen, men utan att röja sitt innehåll. En lämplig max-storlek kan väljas för att undvika onödiga problem.
- Upprepa: Låta användaren ange en ruta att gå till.

  1 Om rutan innehåller en mina, avbryts programmet, Osquar sprängs.

  2 Om rutan är tom och gränsar till minst en mina, skrivs antal angränsande minor ut i rutan på skärmen.

  3 Om rutan är tom och inte gränsar till någon mina, så visas en tom ruta.

  tills minfältet är röjt (dvs alla icke-minor visade) eller Osquar död.
- Visa hela planen då spelet är slut.

Här kan du få idéer om hur programmet skall fungera:

- På Windows: Titta på (och kör) programmet MSröj.
- På CSC:s UNIX-datorer: Starta MatLab, ge kommandot expomap, välj alternativet games och spelet bombs

Gör sedan en tio-i-topp-lista över bästa spelare hittills. Listan ska lagras på en fil mellan körningarna. Du måste nu införa något mått på skicklighet. Ett krav är att tiden ska vara med. Läs i filen `www.csc.kth.se/~lk/P/tidochdatum.txt` om tidavläsning.

Tips: I filen `www.csc.kth.se/~lk/P/random.txt` får du tips till slumpfunktionen, som du använder till att slumpa fram minornas positioner.

Minfältet kan lämpligen representeras av en matris. För varje ruta behöver du lagra huruvida den innehåller en bomb eller ej. Det kan också vara praktiskt att veta om den är röjd än. Tänk efter om du behöver ytterligare information!

På skärmen kan ett tecken ge information om en ruta på minfältet:

0 eller blank Ingen mina på denna ruta. Ej heller runt omkring.

1 - 8 Ingen mina på denna ruta, men (1-8) angränsande minor.

M Här ligger det en mina.

***** Den här rutan är inte undersökt av Trofast än.

Givetvis skall ingen spelare kunna fuska, men när du testar ditt program kan en fuskprocedur som visar var minorna ligger var bra att ha.

Extrauppgift, betyg C: Inför felkontroll av användarens inmatning.

Extrauppgift, betyg B: Ge Osquar (användaren) möjlighet att sätta 'flaggor' där han räknat ut att det finns minor. På så vis slipper han ju tänka ut det igen. Har han tänkt rätt, är ju faktiskt minan upptäckt. Spelet kan nu vinnas genom att alla tomma rutor är öppnade eller att alla minor (och inget annat än minor) är flaggade. Om Osquar går på en mina, ska han få veta hur många minor som röjts (d v s är flaggade) Se till att det finns möjlighet att öppna också en flaggad ruta (man kan ju komma på att man tänkt fel).

Om man klickar på en ruta och den är tom och inte gränsar till någon mina, visas hela det sammanhängande område med tomma rutor som rutan ingår i samt kanten på detta område bestående av rutor med siffror (enligt 2).

Problemet löses enklast med hjälp av rekursion, alltså att man skriver en funktion som anropar sig själv.

Extrauppgift, betyg A: Gör ett grafiskt användargränssnitt så att man kan klicka på rutorna istället för att ange koordinater.

CSC, KTH
DD1314 våren 2017 (Python)

172 Hungriga huggormar

P-uppgiften ska göras individuellt. Läs CSC:s hederskodex innan du börjar!

Varudeklaration: Datastrukturer, slumptal, grafik

Hilda och Hilding Huggorm är två små, nykläckta ormyngel. Platsen dom lever på är den lilla fyrkantiga Ormön där dom börjar äta grodlår som ormar brukar göra. En groda ökar deras längd med en längdenhet. Födotillgången är varierande ; som mest kan H. och H. sätta i sig 3 grodor åt gången. Ibland kanske de bara får tag på en groda.

För att inte tära för mycket på grodtillgången (dåliga grodår = inga grodlår) får inte Hilda och Hilding komma för nära varandra: Gör dom det så slutar dom att växa! Det är naturligtvis inte bra om dom slår knut på sig själv i sitt sökande efter föda. H. och H. är därför noga med att inte korsa sig själva när dom växer och växer och växer...

Ön vid spelets början ser kanske ut så här (i ett separat grafikfönster):

```
* = Här ligger Hilding nykläckt och hungrig.
+ = Här ligger Hilda med lika glupande aptit.
tom ruta = Ingen orm här.
```

	1	2	3	4	5	6	7	8
A								
B								
C						*		
D								
E								
F								
G			+					
H								

Efter att ha ätit två portioner var kan de ha vuxit ut så här:

- Hilda äter 3 grodor och Hilding 2.
- Hilda äter 1 groda och Hilding 3.

```
* = Hilding som först vuxit norrut och sedan västerut.
+ = Hilda som vuxit österut och sedan norrut.
tom ruta = Ingen orm här.
```

	1	2	3	4	5	6	7	8
A			*	*	*	*		
B						*		
C						*		
D								
E								
F						+		
G			+	+	+	+		
H								

Din uppgift är att skriva ett program som låter två spelare anta Hildas och Hildings roller. Programmet slumpar fram en portion grodor, säg 1 - 3 st, som Hilda äter upp. Programmet frågar Hilda åt vilket håll hon vill växa. Hilda svarar då ett av några möjliga alternativ (hur många då?) Om hon då slår knut på sig själv är spelet över. Likaså om hon inte kan växa åt något håll (om hon då skulle växa ut i vattnet eller korsa Hilding).

Därefter serveras Hilding en portion och programmet upprepar allt som står ovan för Hilding.

Olika personer kan vilja spela spelet med olika inställningar, t.ex. hur många spelare som ska köra samtidigt, namnet på ormarna/spelarna och hur stor planen ska vara så ska du spara dessa inställningar på fil. Antalet spelare kommer i C delen.

Det här skall ditt program göra:

1 Slumpa fram en veckoranson grodlår till Hilda.

2 Hilda (spelare 1) väljer sedan i vilken riktning hon väljer att växa.

3 Upprepa 1) - 2) ovan för Hilding (spelare 2).

4 Om någon av ormarna inte kan växa vidare är spelet slut. Programmet skall då tala om hur långa Hilda och Hilding blivit. Den orm som förorsakade avbrottet förlorar (storleken har som bekant inte någon betydelse). Om ingen av ormarna kan växa vidare avbryts också spelet och den som är längst vinner.

Extrauppgift, betyg C: Se till att användarens inmatning kontrolleras. Lägg också till att man kan köra flera spelare, antalet ska anges i filen och alla ska ha ett eget namn.

Extrauppgift, betyg B: Utöka programmet så att användare 1 (Hilda) kan välja om hon skall spela mot datorn eller mot en annan användare. Låt Hilding (om datorn spelar Hilding) ha en strategi för hur han växer bäst. Denna bör ge intelligenta Hilda (som är mån om att kroppen skall vara lång, slank och otrasslig) en god chans att vinna.

Extrauppgift, betyg A: Gör ett grafiskt användargränssnitt så att användaren kan klicka i den riktning ormen ska växa.

CSC, KTH
DD1314 våren 2017 (Python)

174 Patiensen Klockan

P-uppgiften ska göras individuellt. Läs CSC:s hederskodex innan du börjar!

Varudeklaration: Datastrukturer, grafik (i första extrauppgiften).

Patiensen Klockan går till så här: Man blandar en vanlig kortlek med 52 kort. Sedan lägger man ut de tretton översta korten i leken i en rad på bordet. Alla kort läggs med framsidan upp, så att man kan se deras valörer.

Nu gäller det att få ett ess överst i den första högen, en tvåa överst i den andra o s v. I elfte högen ska en knekt ligga, i tolfte en dam och i trettonde en kung. Till sin hjälp har man resten av kortleken. Man fortsätter att lägga kort i samma ordning som förut med skillnaden att om rätt valör redan ligger på en plats hoppas denna över. De nya korten täcker delvis de gamla. Patiensen går ut om man lyckas få rätt valör överallt, och misslyckas om högen tar slut utan att patiensen gått ut.

Du ska göra ett program som blandar en kortlek och lägger Klockan. Visa en rad på skärmen för varje varv du lägger i Klockan. En omgång (som inte gick ut) kan se ut så här:

```
KLOCKAN:

K2  R9  S5  K6  KD  RT K3  KK  HK  HD  R2  S6  RK
SD  H9  RJ  ST  H3  SK R7  HT  K8  S2  H6  R8
SA  HA  R3  H7  HJ  H4     S3  R4  K7  K9  RD
    K5      S4  S8  S7     S9  SJ  H5  H2
    H8          RA  KA     K4  R5  R6  KT
    KJ

Högen tog slut utan att patiensen gick ut. Låt knekt representeras av
   bokstaven J, som i engelska ''jackets'' och 10 med bokstaven T.
```

Här ser man att ruter kung hamnade på rätt plats redan i utlägget. Sedan lades inga fler kort på plats tretton. Ruter sju hamnade rätt på andra utlägget osv. Högen tog slut med klöver knekt. Därför är inga fler kort lagda i sjätte raden. Programmet kommenterar resultatet.

Ditt program ska låta användaren välja mellan:

- Se utlägg av en patiens.
- Se statistik på valt antal patienser (dvs användaren anger ett antal, programmet lägger så många patienser utan att visa utlägg på skärmen och berättar sedan hur många som gick ut)
- Sluta (q)

Extrauppgift, betyg C: Inför felhantering. Om användaren matar in ett felaktigt val ska programmet meddela detta och be om en ny inmatning. Om användaren väljer statistikvalet ska programmet kontrollera att det är ett heltal som användaren matar in, om så inte är fallet ska programmet be om ett nytt värde.

Extrauppgift, betyg B: Du har säkert upptäckt att Klockan inte går ut speciellt ofta. Modifiera ditt program så att när ett kort hamnar på rätt plats på bordet, hamnar de felaktiga kort som låg där innan sist i högen av olagda kort. Dvs när Ruter sju läggs i exemplet ovan, hamnar klöver tre sist i högen av olagda kort. När spader ess läggs, hamnar klöver två näst sist och spader dam sist i högen. (OBS ordningen!) De kort som hamnar sist i högen läggs på så vis ut en gång till på bordet. Egentligen är detta den riktiga versionen av Klockan. Jämför statistikresultaten på den här versionen med resultaten på den gamla!

Extrauppgift, betyg A: Gör ett grafiskt gränssnitt som visar hur klockan ser ut när man lägger den.

När man lägger Klockan med riktiga kort, brukar man i allmänhet lägga korten i form av en urtavla (därav namnet). Esset hamnar då förstås på ettans plats, tvåan på tvåans och så vidare runt till damen på tolvans plats. Kungen får ligga mitt i urtavlan. Inför grafik i ditt program så att utskriften av patiensen blir i form av klockor i stället. Man vill ju fortfarande kunna se alla steg i patiensen, så en klocka per varv måste visas. Ej nylagda kort i varje varv ska vara med, men man ska tydligt se vilka som är nylagda och vilka som inte är det. Exemplet ovan kan då se ut så här:

```
KLOCKAN:

       S6                  R8                  RD
    R2    K2              H6    SD              K9    SA
  HD        R9          S2        H9          K7        HA
 HK    RK    S5        K8   (RK)   RJ        R4   (RK)   R3
  KK        K6          HT        ST          S3        H7
    K3    KD              R7    H3              (R7)    HJ
       RT                  SK                  H4

      (RD)                (RD)                (RD)
    H2   (SA)            KT   (SA)            (KT) (SA)
  H5        K5          R6        H8          (R6)       KJ
 SJ   (RK)  (R3)       R5   (RK)  (R3)       (R5)  (RK)  (R3)
  S9        S4          K4       (S4)        (K4)      (S4)
   (R7)   S8              (R7)   RA              (R7)  (RA)
       S7                  KA                  (KA)

Leken tog slut utan att patiensen gick ut.
```

CSC, KTH
DD1314 våren 2017 (Python)

178 Periodiska systemet

P-uppgiften ska göras individuellt. Läs CSC:s hederskodex innan du börjar!

Varudeklaration: Filhantering, datastrukturer, sortering.

Du ska skriva ett träningsprogram för periodiska systemet. Vid körning kan det se ut så här.

```
-------------- MENY ---------------
1. Visa alla atomer
2. Träna på atomnummer
3. Träna på atombeteckningar
4. Sluta
-----------------------------------
Vad vill du göra? 2


Vilket atomnummer har I ? 35
   Fel svar, försök igen.
   Vilket atomnummer har I ? 53
   Rätt svar!
```

På filen `www.csc.kth.se/~lk/P/avikt.txt` finns alla atombeteckningar med atomvikter lagrade. Kopiera den och studera hur den är upplagd. Programmet ska börja med att läsa in atombeteckningar och atomvikter från filen till en datastruktur. Eftersom atombeteckningarna är sorterade i bokstavsordning måste du sedan sortera datastrukturen efter atomvikt för att få atomnummerordning. Eftersom atomnummerordningen inte exakt följer atomvikterna så måste du dessutom byta plats på följande ämnen:

- Nr 18 mot 19 (Ar mot K)
- Nr 27 mot 28 (Co mot Ni)
- Nr 52 mot 53 (Te mot I)
- Nr 90 mot 91 (Th mot Pa)
- Nr 92 mot 93 (U mot Np)

Programmet ska ge användaren max tre försök på en given fråga, sen ska det rätta svaret visas.

VGV

Extrauppgift, betyg C: Lägg till frågor om atomvikten. Ge användaren tre svarsalternativ att välja mellan, annars blir frågan för svår!

Inför även felkontroll för användarens inmatning.

Extrauppgift, betyg B: Det är egentligen viktigare att lära sig i vilken kolumn en atom finns än att lära sig dess atomvikt utantill. Om du inför rad och kolumn för varje atom kan du låta datorn rita upp ett tomt periodiskt system. Sedan ska programmet be användaren placera in en atom i taget (i slumpvis ordning) på rätt plats tills periodiska systemet är fullständigt.

Extrauppgift, betyg A: Gör ett grafiskt användargränssnitt till programmet!

Datafiler och hjälpfiler: `www.csc.kth.se/~lk/P/avikt.txt`

CSC, KTH
DD1314 våren 2017 (Python)

184 Patiens med annorlunda kortlek

P-uppgiften ska göras individuellt. Läs CSC:s hederskodex innan du börjar!

Varudeklaration: Datastrukturer, extrauppgift med grafik

Den gamla hederliga kortleken med 52 kort byts i den här uppgiften ut mot en nyskapande kortlek med 81 kort. Varje kort har en färg, en symbol, en fyllnadsgrad och ett visst antal symboler. Om man har tre färger, tre symboler, en till tre olika fyllnadsgrader och antalet symboler är en till tre får man efter lite tänkande fram att det går att skapa 81 olika kort. De 81 olika spelkorten ska genereras i programmet.
Sex av dessa kort läggs upp på ett spelbord och sedan ska användaren försöka plocka ut tre kort som har något gemensamt, till exempel färgen, eller så ska de tre korten vara helt olika. Om man tar tre kort där något är lika får man två poäng och om man tar tre kort som är helt olika får man tre poäng. Om man väljer ut tre kort som inte uppfyller något av kraven straffas man med en poängs avdrag.
En textversion av spelbordet kan se ut som nedan:

Nr.	Färg	Fylln.	Symbol	Antal
1	blå	1	triangel	1
2	gul	3	triangel	2
3	gul	3	kvadrat	3
4	blå	3	triangel	3
5	gul	1	triangel	1
6	blå	3	kvadrat	1

Användaren ska sedan ange vilka kort som ska väljas ut. Programmet ska undersöka om de utvalda korten uppfyller kriterierna för att få poäng och sedan addera dessa till den tidigare poängsumman. När detta är gjort tas tre nya kort ur leken och placeras på de platser på spelbordet där de tre tidigare korten låg.
Efter varje runda skrivs den utdelade poängen, total poäng och hur många kort som är kvar i leken ut. Dessutom ska användaren tillfrågas om spelet ska fortsätta eller om man är nöjd med den poäng man har.
Spelet avslutas av sig själv när kortleken tagit slut. Då skrivs den erhållna poängen ut.

Lägg värdena för de olika poängen och avdragen man kan få i en fil så att det är lätt att ändra på utan att behöva gå in i koden.

Tips: Gör en ordentlig struktur med klasser och fundera med hjälp av papper och penna på hur korten ska genereras.

Tips: Exempel på färger och symboler: gul, blå, röd, triangel, kvadrat och cirkel

Extrauppgift, betyg C: Gör så att programmet kontrollerar att man har valt ett kort i intervallet 1...6 och att samma kort inte har valts två gånger. Sekvensen 1 3 3 ska alltså innebära att användaren får ett felmeddelande och ombeds göra en ny inmatning.

Extrauppgift, betyg B: Låt datorn lägga patiensen!

Uppfinn en algoritm som givet n kort (där n är en multipel av 3) plockar ut den följd av kort-tripplar som ger mest poäng totalt.

Extrauppgift, betyg A: Gör ett grafiskt gränssnitt där man kan klicka på korten för att välja dem.

CSC, KTH
DD1314 våren 2017 (Python)

187 Aktieköp

P-uppgiften ska göras individuellt. Läs CSC:s hederskodex innan du börjar!

Varudeklaration: Sortering , filhantering , datastrukturer, flexibilitet.

Programmet skall ge vägledning vid aktieköp. För att ta reda på om aktien är köpvärd kan antingen en fundamental analys utföras eller en teknisk analys. Vid en teknisk analys tittar man på aktiens historik. Teknisk analys använder man sig främst av när man skall ha aktien en kort tid . Fundamental analys är viktigare när man skall behålla aktien en längre tid. Programmet skall också kunna rangordna aktier efter deras betavärde. Använd inte bara de aktier som finns utan utöka med minst en till aktie där du hittar på någon intressant data.

Så här ska programmet se ut:
——————————Meny——————————
1.Fundamental analys (Vid långsiktigt aktieinnehav)
2.Teknisk analys (Vid kort aktieinnehav)
3.Rangordning av aktier med avseende på dess betavärde
4.Avsluta
Vilket alternativ vill du välja? 1

En fundamental analys kan utföras för följande aktier:
1.Ericsson
2.Electrolux
3.AstraZeneca
Vilken aktie vill du göra fundamental analys på? 1

————————Fundamental analys för Ericsson————————
företagets soliditet är 36 %
företagets p/e-tal är negativt
företagets p/s- tal är 0.23

Vilket alternativ vill du välja? 2
En teknisk analys kan utföras för följande aktier:
1.Ericsson
osv...
Vilken aktie vill du göra teknisk analys på? 3

————————Teknisk analys för AstraZeneca————————- (*1)
kursutveckling(30 senaste dagarna) 4,4 %
betavärde 1,25
lägsta kurs(30 senaste dagarna) 502
högsta kurs(30 senaste dagarna) 509
Vilket alternativ vill du välja? 3

——Rangordning av aktier med avseende på dess betavärde—— (*2)
1.Ericsson 1,6
2.AstraZeneca 1,25
3.Elektroux 1,1

*1)Här läser man in värden från 'kurser.txt' och beräknar kursändringen (i procent) på de senaste 30 dagarna. Man beräknar även betavärdet för aktien från värdena i textfilen. Man beräknar också högsta och lägsta kurs under perioden.

*2)Rangordningen går till så att man sorterar efter det betavärde som man räknat ut vid den tekniska analysen.

Man hämtar underlag från 3 filer, den första är 'fundamenta.txt'. Där ska det finnas fundamentala data om de olika aktierna.
Format: företagsnamn, soliditet, p/e, p/s
Ericsson
30
negativt
0.35
Electrolux
40
10
0.40
En annan fil man behöver är 'kurser.txt' där man hittar teknisk data för att göra en teknisk analys. Kurser från de 30 senaste dagarna är med.
Format: datum, börskurs
Ericsson
02-08-01 7.15
02-08-02 6.72
...
Electrolux
02-08-01 167.50
02-08-02 165.50
...
Man behöver även en textfil med börsens omx-index 'omx.txt' för att kunna räkna ut betavärdena. Betavärdet = avkastning på aktien / avkastning för marknaden under en viss period. Med avkastning för en viss period menas värdet vid periodens slut / värdet vid periodens början (för ett värdepapper eller index).
Format: datum, index
02-08-01 529.93
02-08-02 519.94

Extrauppgift, betyg C: Inför fullständig felhantering av användarens inmatning.

Extrauppgift, betyg B: De aktiedata som medföljer denna uppgift är endast av historiskt intresse. Hitta eller skapa en webbsida med aktuell aktieinformation och läs från den.

Extrauppgift, betyg A: Gör ett grafiskt användargränssnitt (GUI) till programmet.

Datafiler och hjälpfiler: `www.csc.kth.se/~lk/P/fundamenta.txt`, `www.csc.kth.se/~lk/P/kurser.txt`, `www.csc.kth.se/~lk/P/generalindex.txt`

CSC, KTH
DD1314 våren 2017 (Python)

192 Kalender

P-uppgiften ska göras individuellt. Läs CSC:s hederskodex innan du börjar!

Varudeklaration: Datastrukturer, stränghantering

Som programledare i det populära TV-programmet "Dejten" har du ett hektiskt medieliv med möten med nya och gamla deltagare, inspelningar och inte minst pressen. Din vardag måste organiseras stenhårt. Gör ett program som fungerar som en kalender. Varje "sida" är en textrad med en minnesanteckning av typen
"`2010-10-31: Träffa de nya deltagarna inför avsnitt 137`"
eller
"`2011-01-03: Träffa Svante Djupsund och berätta att han blivit spolad från serien.`".

Du ska minst ha dessa alternativ i din meny:

```
1. Bläddra framåt
2. Bläddra bakåt
3. Sätt in ny sida
4. Ta bort sidan
5. Visa alla sidor
6. Avsluta
```

När man sätter in en ny sida ska den placeras in efter datum. Man vill inte behöva bläddra förbi en massa tomma sidor, så alternativen 1 och 2 ska hoppa direkt till nästa datum där det finns en minnesanteckning.

En annan sak som är viktig är att informationen i kalendern sparas på fil. Man kan fritt välja om man vill spara allt i samma fil eller om man vill ha en fil per dag.

Extrauppgift, betyg C: Inför felhantering för menyalternativ (vid felaktigt menyval ska programmet fråga om). Se till att det datum användaren anger för en ny sida verkligen är ett giltigt datum.

Extrauppgift, betyg B: Skapa en färdig kalender över helt år med datum från början. Inför möjligheten att ha flera aktiviteter per dag, och att man kan ändra och ta bort enskilda aktiviteter från en viss dag. Man ska också kunna kolla hela månadens aktiviteter.

Extrauppgift, betyg A: Gör ett grafiskt användargränssnitt där man kan klicka för att bläddra sig fram i kalendern. Aktuell sida vid start ska vara dagens datum.

Datafiler och hjälpfiler: `www.csc.kth.se/~lk/P/tidochdatum.txt`

CSC, KTH
DD1314 våren 2017 (Python)

193 Memory

P-uppgiften ska göras individuellt. Läs CSC:s hederskodex innan du börjar!

Varudeklaration: Datastrukturer, inläsning från tangentbord och filhantering.

Du ska skriva ett program som spelar Memory. Användaren ska försöka lösa en A x A (6 x 6 i exemplet) matris genom att matcha ord i olika celler i en matris. Så här kan det se ut:

```
   1   2   3   4   5   6
A --- --- --- --- --- ---
B --- --- --- --- --- ---
C --- --- --- --- --- ---
D --- --- --- --- --- ---
E --- --- --- --- --- ---
F --- --- --- --- --- ---
==============================

val1: A5

   1   2   3   4   5   6
A --- --- --- --- kul ---
B --- --- --- --- --- ---
C --- --- --- --- --- ---
D --- --- --- --- --- ---
E --- --- --- --- --- ---
F --- --- --- --- --- ---


val2: F6
   1   2   3   4   5   6
A --- --- --- --- kul ---
B --- --- --- --- --- ---
C --- --- --- --- --- ---
D --- --- --- --- --- ---
E --- --- --- --- --- ---
F --- --- --- --- --- ful

==============================

val1: D1

   1   2   3   4   5   6
A --- --- --- --- --- ---
B --- --- --- --- --- ---
C --- --- --- --- --- ---
D kul --- --- --- --- ---
E --- --- --- --- --- ---
F --- --- --- --- --- ---
```

```
val2: A5

   1   2   3   4   5   6
A --- --- --- --- kul ---
B --- --- --- --- --- ---
C --- --- --- --- --- ---
D kul --- --- --- --- ---
E --- --- --- --- --- ---
F --- --- --- --- --- ---

du klarade 1 par.
==================================

osv....
```

Programmet uppmanar användaren att välja två celler. Varje gång användaren väljer en cell skrivs hela matrisen ut på skärmen där ordet i cellen blir synlig (kolla bilden). Sedan kontrollerar programmet om de två valda celler innehåller samma ord och i så fall visas båda orden när matrisen skrivs ut under resten av programmet. Orden göms självklart i fall de inte är samma.
Filen memo.txt innehåller 732 ord på var sin rad och varje ord består exakt av 3 bokstäver. Programmet ska slumpa 18 ord från filen, kopiera varje ord och placera ut orden i en matris som sedan används av programmet. Programmet avslutas när användaren har lyckats visa alla ord i matrisen. Vid avslutningen skriver programmet ut det antal försök användaren gjort och visar hans placering på ett highscore som ska sparas på fil.

Extrauppgift, betyg C: Inför felhantering för användarens inmatning.

Extrauppgift, betyg B: Gör så att det går att använda ord av varierande längd. Ordlängden ska förstås inte framgå av bilden. Detta kan enklast fixas genom att man sätter antalet streck till längsta ordet.

Extrauppgift, betyg A: Gör en grafisk version av spelet.

CSC, KTH
DD1314 våren 2017 (Python)

195 Zoo

P-uppgiften ska göras individuellt. Läs CSC:s hederskodex innan du börjar!

Varudeklaration: Datastrukturer, inläsning från tangentbord och filhantering, tid och datum.

Du ska skriva ett program som vägleder besökare till ett zoo. Besökaren ska kunna mata in datum och tidsintervall för besöket, och programmet ska skriva ut vilka djur som förväntas vara vakna, och när dessa djur matas (om matningen infaller under besöket). Så här kan det se ut:

```
Vilket datum vill du besöka Neptunus Zoo? 21 juni
Zooet är öppet mellan 06-23.
Vilken tid vill du komma? 13-16
Under ditt besök kan du se:
Björn
Sjölejon  *** matas kl 14 ***
Säl       *** matas kl 14 ***
Varg
Älg
```

Information om djurens vakentider ska finnas lagrad på fil, t ex på följande format

```
Format:namn / ide / vakentid / matningstid
Björn / vinter / 9-20 / 12
Latmask / sommar / 12-14 / 13
Nattuggla / - / 21-05 / 21
Sjölejon / - / 6-18 / 14
Säl / - / 6-18 / 14
Varg / - / 6-20 / 12
Älg / - / 7-19 / 10
```

Extrauppgift, betyg C: Inför felkontroll för all inmatning.

Extrauppgift, betyg B: Låt besökaren ange vilka djur hen vill se. Ditt program ska då föreslå ett datum och en tid då alla dessa djur kan beskådas (om det är möjligt).

Extrauppgift, betyg A: Gör ett grafiskt gränssnitt till programmet. Dagens schema ska visas upp när programmet startar, men man ska kunna klicka sig fram till andra datum.

Datafiler och hjälpfiler: `www.csc.kth.se/~lk/P/tidochdatum.txt`

CSC, KTH
DD1314 våren 2017 (Python)

198 Salladsbaren

P-uppgiften ska göras individuellt. Läs CSC:s hederskodex innan du börjar!

Varudeklaration: Datastrukturer, inläsning från tangentbord och filhantering.

Din kompis ”Kalle på hörnet” driver en vegetarisk restaurang som specialiserar sig på sallader. Han har bett dig att skriva ett litet program där folk får välja och vraka bland alla deras olika alternativ.

Programmet ska ha följande steg:

- Skriv ut vilka ingredienser som finns att tillgå.
- Kunden ska ange vad hen vill ha i sin sallad.
- Om det finns en eller flera sallader som stämmer överens med de valda ingredienserna så ska alla dessa visas.
- Om det inte finns någon sallad som matchar så ska den sallad som har flest matchande ingredienser föreslås tillsammans med en lista över vilka ingredienser som behöver kompletteras och totalkostnaden. Finns det flera sallader som kan komma ifråga ska den billigaste väljas.
- Efter att kunden har valt sin sallad visas menyn för extra val där alla ingredienser och deras priser listas.
- När kunden är nöjd så ska ett kvitto skrivas ut på fil och på skärmen.

När programmet startar så ska du läsa in alla sallader från en fil. Det som ska finnas med för varje sallad är ett namn, pris och ingredienser. Utöver detta så får du strukturera filen hur du vill.

Det ska också finnas en separat fil där alla ingredienser står var för sig med ett pris. Denna data laddas in och används sen när man ska lägga till enskilda ingredienser till salladen.

Extrauppgift, betyg C: Inför felhantering av användarens inmatning.

Extrauppgift, betyg B: Du ska nu göra en huvudmeny där du ska kunna välja bland flera olika salladsbarer. Inget av datan för vilka restauranger som finns ska finnas i programmet utan allt ska finnas på fil.

Extrauppgift, betyg A: Gör ett grafiskt användargränssnitt genom vilket all interaktion sker.

CSC, KTH
DD1314 våren 2017 (Python)

105 Arga Troll

P-uppgiften ska göras individuellt. Läs CSC:s hederskodex innan du börjar!

Varudeklaration: Datastrukturer, inläsning från tangentbord och filhantering.

Som vi alla vet så finns det många troll som bor i skogen och vissa av dessa troll kan vara ganska arga. Det är därför viktigt att trollen inte kan se några andra arga troll för då börjar de bråka. Detta är inte ett lika stort problem som man skulle kunna tro då troll är ganska dumma. De är väldigt bundna till naturen och tittar bara i de fyra väderstrecken samt diagonalerna mellan väderstrecken, sydöst,sydväst, osv.

Du ska göra ett spel där man ska placera ut arga troll på ett fyrkantigt bräde. Den som spelar spelet ska själv få välja hur stort brädet är (upp till en rimlig max-storlek).

Reglerna för spelet är ganska enkla, det ska finnas:

- Ett troll per rad.
- Ett troll per kolumn.
- Inga troll får finnas på samma diagonal.

När man väl startat programmet så ska följande saker hända:

- Spelet börjar med att man hälsas välkommen till programmet och reglerna förklars.
- Spelaren får välja storleken på bräde. En maxstorlek på 8 kan införas för att försäkra att spelet förblir spelbart.
- Spelaren får sedan börja placera ut trollen en taget, rad för rad. Spelaren ska kunna få ångra sitt val genom att skriva ”undo”i stället för ett kolumnnummer.

Du ska även ta tid för hur lång tid det tar för spelaren och lägga in det i en highscore. Placeringen ska både vara baserad på tid och storlek på brädet.

Extrauppgift, betyg C: Inför felhantering för användarens inmatning.

Extrauppgift, betyg B: Du ska nu skriva en algoritm som löser problemet för små bräden, minst av storlek 5 (även om den borde klara av även 6 och 7). Algoritmen går ut på att man försöker att placera ut trollen rad för rad och om man misslyckas så går man tt-illbakaöch testar nästa alternativ. Det är viktigt att vara medveten om att det tar väldigt mycket mer tid att lösa ett bräde som är ens en storlek större.

Extrauppgift, betyg A: Du ska göra ett GUI där spelaren får placera och tar bort troll från brädet genom att klicka på rutorna.

CSC, KTH
DD1314 våren 2017 (Python)

116 Djurparken

P-uppgiften ska göras individuellt. Läs CSC:s hederskodex innan du börjar!

Varudeklaration: Datastrukturer, inläsning från tangentbord och filhantering.

En djurälskande släkting har vunnit på lotto och köpt sig en liten djurpark! Som den trevliga teknologen du är erbjuder du dig att skriva ihop ett program som hjälper din släkting att hålla koll på de nyinköpta djuren.

Det första du ska göra är att skapa en fil där alla djur ligger. De parametrar du måste ha med är Namn, Ålder, Art och Kön. Det ska vara minst 7 djur och det måste finnas minst två som är av samma art. Tänk på att programmet blir mycket roligare desto fler djur du har.

Man ska kunna göra följande i programmet:

- Söka efter ett djur baserat på parametrar, t.ex. namn, ålder, art, osv.
- Sortera djuren baserat på parametrar, både stigande och fallande.
- Lägga till nya djur i djurparken.
- Sälja (ta bort) bråkiga djur från djurparken.

Här är det mycket viktigt att informationen presenteras på ett snyggt sätt så att användaren lätt kan dra slutsatser om datan som presenteras.

När programmet avslutas så sparas alla ändringar tillbaka till filen.

Extrauppgift, betyg C: Inför felhantering för alla inputs till programmet. Du ska även se till att man inte råkar döpa två djur till samma sak då en sann djurälskare ger ju alla djuren i parken olika namn.

Extrauppgift, betyg B: Du ska nu införa en maxstorlek för djurparken vilket är ett rekommenderat högsta antal djur som får plats i djurparken. Detta bör då också sparas på fil. Användaren ska nu få rekommendation på vilka djur som borde köpas in eller säljas. Rekommendationerna ska vara baserade både på vad som är bra för djuren och vad ägaren antagligen vill uppnå. Ett minimum är följande saker:

- Man ska få rekommendationer att köpa ett till djur för de djuren som är ensamma (bara en av samma art).
- Om det finns plats kvar i djurparken så ska man få rekommendationer för vilka djur man behöver köpa in för att kunna avla på dem (en av varje kön).
- Har man för många djur ska man få rekommendationer för vilka djur man kan ta bort. Hänsyn ska tas för att se till att djuren inte blir ensamma.

Extrauppgift, betyg A: Du ska nu implementera ett grafiskt interface för djurparken. Använd dig inte bara av text-rutor utan låt en rullgardin eller liknande populeras av de olika djuren som finns i programmet.

CSC, KTH
DD1314 våren 2017 (Python)

111 När vi har tid

P-uppgiften ska göras individuellt. Läs CSC:s hederskodex innan du börjar!

Varudeklaration: Datastrukturer, inläsning från tangentbord och filhantering.

När flera personer ska träffas samtidigt så är det inte alltid lätt att hitta den tid som passar bäst. Därför har du bestämt dig för att knåpa ihop ett finurligt lite program som löser det här problemet. Du ska skapa ett program som låter flera personer välja när de kan träffas.

Programmet består av fyra delar. Skapandet av träffar, angivelserna när man kan, summeringen och avslut.

Delarna ska se ut som följande:

- När man skapar en träff så får man ange två saker, namn på träffen och föreslagna tider (datum och tid) när man eventuellt ska träffas.
- När man ska ange när man kan så får man först ange namn och sen får man välja ett tillfälle som det hela gäller för. Efter det så får blir man tillfrågad om man kan vid alla angivna tillfällen men en ja/nej fråga.
- När man väljer summeringen och har valt träff så ska programmet räkna ut vilken tid som passar bäst och skriva ut alla som kunde träffas vid just det tillfället.
- När programmet avslutas skall all data sparas till fil så att vid nystart kan nya och gamla personer lägga till ny information till schemat.

Extrauppgift, betyg C: Inför felhantering av användarens inmatning.

Extrauppgift, betyg B: Du ska nu införa ja/nej/kanske istället för bara ja/nej som val. Du ska dessutom lägga till en ägare för varje träff och bara den personen ska kunna genomföra en summering (ingen säkerhet krävs i programmet).

Extrauppgift, betyg A: Gör ett grafiskt användargränssnitt för programmet.

CSC, KTH
DD1314 våren 2017 (Python)

107 Reversi

P-uppgiften ska göras individuellt. Läs CSC:s hederskodex innan du börjar!

Varudeklaration: Datastrukturer, inläsning från tangentbord och filhantering.

Reverse är ett spel där två spelare turas om med att placera ut svarta respektive vita brickor på ett fyrkantigt bräde. Innan spelet börjar skall användaren välja spelplanens storlek (upp till rimlig storlek). När spelet är slut ska antalet mörka och ljusa brickor skrivas ut på skärmen samt vem som vann. Du kan implementera valfritt sätt att välja vem som börjar.

Regler:

- Spelarna turas om att lägga varsin bricka.
- Måste lägga bricka så att man vänder på något (går det ej står man över en tur).
- Riktningar som vänder är horisontellt, diagonalt och vertikalt.
- Allt som kan vändas av ett drag vänds.

Startläge: Startläget för spelet är alltid fyra st brickor i mitten av brädet som ligger på varsin diagonal.

Spelet avslutas när antingen brädet är fullt eller när båda spelarna har tvingats att stå över sitt drag då de inte kan ta någon bricka.

Du ska även räkna med hur mycket en person vann och lägga in det i en highscore som sparas på fil.

Extrauppgift, betyg C: Inför felhantering av användarens inmatning.

Extrauppgift, betyg B: Du ska nu låta datorn spela. Låt datorn ta så många pjäser som det går i varje drag. Glöm inte att datorn måste kunna stå över drag när så krävs.

Extrauppgift, betyg A: Du ska nu göra ett grafiskt interface där spelaren gör sitt drag genom att klicka på den ruta han vill lägga sin pjäs. Missa inte någon av den tidigare funktionaliteten för spelet.

CSC, KTH
DD1314 våren 2017 (Python)

106 Damspel

P-uppgiften ska göras individuellt. Läs CSC:s hederskodex innan du börjar!

Varudeklaration: Datastrukturer, inläsning från tangentbord och filhantering.

Dam, damspel, är ett strategiskt brädspel som spelas av två spelare på ett kvadratiskt bräde där användaren själv får välja storlek så länge det är en multipel av 2 och rimligt stor (en begränsning på 08-12 rekommenderas).

Dam spelas av två personer. Det finns en rad olika regeluppsättningar och damvarianter. Gemensamt för dem alla är att man startar med sin halva av planen (minus raden närmast motståndaren) fylld med en pjäs på varannan ruta i form att ett schackmönster. Spelarna turas om med att göra drag varsitt drag. Spelaren ska få välja sin egen symbol. Målet är att fånga motståndarens brickor genom att hoppa över dem.

Reglerna för att göra ett drag är följande:

- Brickorna får bara hoppa diagonalt.
- Brickorna får bara hoppa en ruta per drag, om man inte tar en av motståndarens brickor.
- Man tar en bricka genom att hoppa över den. Brickor på kanten är därför säkra.

Den som mister alla sina brickor har förlorat.

Om spelarna väljer att avsluta innan någon har förlorat ska det aktuella spelläget (spelplanen) sparas på fil, så att man ska kunna välja att fortsätta spelet från samma läge nästa gång programmet startas.

Extrauppgift, betyg C: Inför följande regler:

- Man måste ta motståndarens bricka om man kan och även fortsätta att ta så länge som det går.
- En bricka får bara hoppa framåt såvida den inte har nått hela vägen fram till motståndarens sista rad. Då får den hoppa baklänges också.
- Kontrollera att användarens inmatning är giltig.

Extrauppgift, betyg B: Du ska nu låta datorn spela. låt datorn ta så många pjäser som det går och om datorn inte kan ta något så slumpar den ett drag.

Extrauppgift, betyg A: Du ska nu göra ett grafiskt interface där spelaren gör sitt drag genom att klicka på pjäsen och sen klickar på vart han vill flytta den. Missa inte någon av den tidigare funktionaliteten i spelet.

CSC, KTH
DD1314 våren 2017 (Python)

120 Parkeringshuset

P-uppgiften ska göras individuellt. Läs CSC:s hederskodex innan du börjar!

Varudeklaration: Filhantering, datastrukturer, sortering.

Ett parkeringshus har bestämt sig för att bygga ett nytt system för sina trognaste kunder för att underlätta betalningarna och förhoppningsvis få dem att parkera oftare. Parkeringshuset vill också vara miljövänliga så man har bestämt sig för att ta betalt beroende på hur stor bil folk har.

I kundsystemet så har man följande information som sparas på fil mellan programkörningarna: Registreringsnummer, biltyp (liten, mellan eller stor) samt ägaren av bilen. Ett prisexempel kan vara att små bilar kostar 20kr/h, mellanstora 25kr/h och stora kostar 30kr/h.

Vid programkörning ska man från huvudmenyn kunna välja att mata in information om när bil kom till och lämnade parkeringshuset, eller att kunna välja att läsa in en redan existerande fil med tidigare inmatade in- och utfarter. Då programmet avslutas ska alla inlästa och inmatade in- och utpassager skrivas ner på fil, så att de kan läsas in nästa gång programmet körs.

Parkeringshuset har valt att avrunda upp alla parkeringstider till närmast halvtimme för enkelhets skull. Om kunden ifrågasätter priset, så ska även historiken för en bil kunna visas!

Exempel på inmatning av parkerande bil:

```
Välkommen till P-(uppgifts)-huset!
   I  In- och ut-passage
   F  Läs in fil med historik
   N  Lägg till ny bil
   P  Räkna ut parkeringskostnad för en bil
   V  Visa parkeringshistorik för en bil
   S  Avsluta.
   >>> I


   Inpassage/Utpassage!
   Ange registreringsnummer:
   >>> AAA123
   Ange starttid:
   >>> 00:42
   Ange sluttid:
   >>> 13:37
   OK!

   ---> ... tillbaka till huvudmenyn
```

Exempel inläsning av fil:

```
Välkommen till P-(uppgifts)-huset!
   I  Inpassage/Utpassage...
   F  Läs in fil med historik
   N  Lägg till ny bil
   P  Räkna ut parkeringskostnad för bil
   S  Avluta.
   >>> F


   Hämta data från fil!
   Ange filnamn:
   >>> inutpassager.txt
   OK! 17 registreringar lästes in!

   ---> ... tillbaka till huvudmenyn
```

Extrauppgift, betyg C: Inför felhantering i programmet, som kollar att filer man angav verkligen finns, och att formaten på inmatade strängar stämmer. Se även till att datum (välj själv på vilken form och visa detta klart och tydligt för användaren) för in och utpassage sparas i loggfilen. Om fel hittas i textfilen ska programmet upplysa användaren om att fel existerar, på vilken rad felet är, samt hoppa över den raden och köra vidare.

Tips: Låt användaren mata in dagens datum vid start av programmet.

Extrauppgift, betyg B: Inför separat inmatning av inpassage och utpassage. Spara det verkliga klockslaget (enligt datorns tid) när bilen kör in, och beräkna parkeringstiden när bilen kör ut. Anta att varje sekund som passerat motsvarar tio minuters parkeringstid. Det ska gå att avsluta programmet mellan inpassage och utpassage (tiden måste alltså sparas på fil). Priset för att stå ett helt dygn ska vara lägre än priset för 24 timmar.

Extrauppgift, betyg A: Lägg till ett grafiskt gränssnitt.

CSC, KTH
DD1314 våren 2017 (Python)

124 Packlistor för resor

P-uppgiften ska göras individuellt. Läs CSC:s hederskodex innan du börjar!

Varudeklaration: Filhantering, datastrukturer, sortering.

Vid olika resor, utflykter och andra tillfällen när man behöver komma ihåg ett större antal föremål än hjärncellerna räcker till för vore det trevligt om det fanns ett program som kunde vara till hjälp för att komma ihåg alla prylar.

Vid körning av programmet ska användaren kunna skapa nya packlistor, och sedan hålla reda på vilka saker som ska packas med för respektive resa genom att lägga till påminnelser till respektive packlista. Man ska även kunna ta bort föremål som skrevs upp felaktigt.

Användaren ska kunna visa alla saker som står på en viss lista genom att söka efter hela (eller delar) av namnet på packlistan. Om flera packlistor hittas skall användaren med ett menyval få välja en dessa, som sedan skrivs ut.

För att veta vilka aktuella packlistor som finns skall programmet kunna skriva ut samtliga listors namn och datum, sorterat efter just datum. Användaren anger ett datum, och alla packlistor som har ett datum lika med eller efter detta skall skrivas ut. Lämnar man datumet blankt visas alla listor i systemet.

När programmet avslutas ska all information skrivas till en textfil som läses in när programmet startar.

Ett förslag på format i textfilen kan vara:

```
datafil.txt
   ------------------------------------------------------------------------
   Spelning Gamla Stan/20150902
   Trummor/Konfetti/Hallonsaft/Karta över Australien/Konstigt instrument
   Maratonlöpning i historiska fotspår/20150101
   Kappa/Skor/Hög hatt/Fickur/Promenadkäpp
   Utflykt till gamla Trollskogen/20150722
   Yllefilt/Engångsgrill/Kol/Batteridriven elfläkt/Kakor/Osynlighetsmantel
```

Förslag på huvudmeny och visning av en lista:

```
Välkommen till Packningsprogrammet!
   N  Ny packlista
   I  Visa innehåll i lista
   S  Lägg till sak till en lista
   A  Visa alla kommande listor
   S  Avsluta.
   >>> I

   Visa lista!
   Vilket namn på lista letar du efter?
   >>> gamla

   Det finns flera listor som matchar din sökning:
   1. Spelning Gamla Stan - 20150902
   2. Utflykt till gamla Trollskogen - 20150722
   >>> 2
```

```
Dessa saker ska packas med!
   * Yllefilt
   * Engångsgrill
   * Kol
   * En liten men effektiv elfläkt(batteridriven)
   * Kakor
   * Osynlighetsmantel

   ... ---> tillbaka till huvudmenyn
```

Förslag på huvudmeny och att skapa en ny packlista:

```
Välkommen till Packningsprogrammet!
   N  Ny packlista
   I  Visa innehåll i lista
   S  Lägg till sak till en lista
   A  Visa alla kommande listor
   S  Avsluta.
   >>> N

   Skapa ny lista!
   Vad ska listan heta?
   >>> Repetition med bandet

   Vilket datum gäller listan?
   >>> 20151011

   Listan skapades!

   ... ---> tillbaka till huvudmenyn
```

Tips: Låt användaren mata in dagens datum vid start av programmet.

Extrauppgift, betyg C: Inför felhantering i programmet, som kollar att filer man angav verkligen finns, och att formaten på inmatade strängar stämmer. Se även till att datum (välj själv på vilken form och visa detta klart och tydligt för användaren) matas in korrekt, och att datumen faktiskt existerar. Om fel hittas i textfilen ska programmet upplysa användaren om att fel existerar, på vilken rad felet är, samt hoppa över hela den packlistan och köra vidare.

Extrauppgift, betyg B: Ändra datastrukturen så att man nu ska kunna markera föremål som packade. Ge användaren möjlighet att se antingen alla föremål i en specifik packlista, eller bara de föremål som inte prickats av och packats med ännu.

Alla föremål ska presenteras i bokstavsordning!

Extrauppgift, betyg A: Lägg till ett grafiskt gränssnitt.

CSC, KTH
DD1314 våren 2017 (Python)

126 Dalek

P-uppgiften ska göras individuellt. Läs CSC:s hederskodex innan du börjar!

Varudeklaration: filhantering, objekt

Doktorn är jagad av en grupp Daleks i en labyrint. Så här ska spelet fungera:

- För varje steg som Doktorn tar, tar alla Daleks ett steg mot Doktorn.
- Om två Daleks krockar, så ligger det kvar en hög med skrot. Doktorn kan inte gå på högen med skrot – Daleks som åker på den försvinner ur spel.
- Doktorn kan teleportera sig med sin Sonic Screwdriver till en slumpmässig ruta.
- Krockar någon Dalek med Doktorn så förlorar spelaren.
- Har alla Dalek kraschat så vinner spelaren.

Spelet utspelar sig på ett rutnät. Vissa rutor är väggar – alla andra rutor är golv. Doktorn finns på en ruta, varje Dalek på en ruta.

Doktorn flyttar sig enligt styr-kommandon från spelaren. Daleks går så rakt mot Doktorn de kan – genom att ta ett steg till någon av de närmaste 8 rutorna.

Om närmaste riktningen går in i en vägg, så kommer Daleken stå stilla (om den ska rakt in i väggen) eller följa väggen (om den ska snett in i väggen).

Labyrinten finns i en fil: filnamnet matas in av användaren för att starta spelet. Filen innehåller en labyrint beskriven som lika långa rader med . för golv, * för vägg, A för Dalek, # för skrothög och D för Doktorn. Exempelvis kan det se ut så här:

```
**************************************************
   *................................................*
   *.............................**********.........*
   *...A..........................*..............A..*
   *........................D.......*..................*
   *.....**********.................**********........*
   *..............*.....................................*
   *..............*....A..........A.....................*
   *.....**********.....................................*
   *................................................*
   **************************************************
```

Extrauppgift, betyg C: Hantera fel i inmatning och i labyrintfilen: om användaren matar in ett fel så ska programmet beskriva felet och be om ny inmatning.

Labyrintfilen är korrekt om:

- Alla rader är lika långa
- Alla tecken är något av . * # D A

Om labyrintfilen inte är korrekt ska programmet beskriva felet och be om en ny labyrintfil.

Extrauppgift, betyg B: Inför en smartare *Supreme Dalek* som kan gå runt väggar. Låt användaren välja svårighetsgrad (det blir svårare med fler Supreme Daleks).

Extrauppgift, betyg A: Skapa ett grafiskt interface, exempelvis med någon av Pythonmodulerna curses, pygame eller tkinter.