EECS, KTH
DD100N prgw hösten 2018 (Python)

127 Platsbokning på SJ

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

Varudeklaration: Filhantering, datastrukturer.

Skriv ett program som hjälper SJ med platsbokning i en tågvagn. 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 för de senaste bokningarna (de som bokats under denna körning).

```
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.

Försök se till att bokade platser hamnar intill varann. Annars 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.

När tid och datum för avgången passerat tas filen bort.

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

EECS, KTH
DD100N prgw hösten 2018 (Python)

140 Sålda biobiljetter

P-uppgiften ska göras individuellt. Läs EECS: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
Man har tre olika avgiftsklasser: vuxna, pensionärer och barn. 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 och dessutom fråga användaren om antal sålda biljetter i varje prisklass för varje biograf. För varje biograf ska skrivas 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.

Var noga med att det ska vara mycket enkelt att öka mängden biografer. Rimligtvis så bör det räcka med att man lägger till en till biograf i filen.

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.

EECS, KTH
DD100N prgw hösten 2018 (Python)

141 Tennismatch

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

Varudeklaration: Datastrukturer, sortering, viss filhantering.

Du ska skriva ett program för att samla statistik från tennismatcher. Användaren får välja två antagonister ur en spelarförteckning. Denna skriver du ihop själv (skapa en textfil med spelarnas data). Den kan se ut så här:

```
Format:
Namn (max 20 tkn) / sannolikhet att vinna sin serve (0-1)
antal vunna matcher / antal spelade matcher
============================================================
B Borg
0.85
5
7
J Näsman
0.12
 ⋮
```

Programmet presenterar urvalet (se nedan) och användaren väljer ut två av spelarna, som får möta varandra i tennismatchen. Användaren får också mata in vem som vann matchen. Sen ska resultatlistan uppdateras och presenteras på nytt. Spelarna ska sorteras med avseende på vinstprocenten. Det kan t ex se ut så här:

Plac	Namn	vunna	spelade	andel vunna
1	B Borg	5	7	0.714
2	J Connors	12	34	0.353
3	J Näsman	0	12	0.000
⋮				

Extrauppgift, betyg C: Kontrollera att infilen existerar och att den innehåller rimliga data.

Extrauppgift, betyg B: Skriv ett program som simulerar en tennismatch i tre set. Matchen delas upp i set, game och bollar. För att avgöra vem som vinner en boll används random. Servar spelare A och $0 < x < p_A$ så vinner A bollen (p_A = sannolikheten att A vinner sin serve). Efter första vinstbollen har man 15 poäng, efter andra 30 och efter den tredje 40. Vidare finns begreppen ”lika” och ”fördel” samt ”game”.

VGV

Vi illustrerar dessa enklast med ett par exempel på poängställningar, (vi kallar spelarna här för “spelare A” respektive “spelare B”):

```
15 - 0, 15 lika, 15 - 30, 30 lika, 30 - 40
game: spelare B
```

Efter ställningen 40 lika, används endast begreppen ”lika”, ”fördel” och ”game” för att beskriva ställningen (det finns alltså inga poängsiffror > 40). “fördel spelare A” betyder att spelare A leder med en boll.

```
30 - 40, 40 lika, fördel spelare A
game: spelare A
```

Gamet är avslutat först efter att en spelare har vunnit, dvs fått 40 poäng och vinner nästföljande boll och motståndaren har < 40 poäng, eller att någon har haft fördel och vinner nästföljande boll. Spelarna servar vartannat game. Den spelare som först har vunnit 6 game och har minst två game mer än motståndaren (6-4, 6-3 etc), vinner setet. Setet fortsätter tills någon av spelarna har två games övervikt. Matchen spelas i tre set. Vem som börjar att serva avgörs av användaren eller av slumpen.
Användaren skall kunna välja hur ofta han vill se ställningen, efter varje boll, efter varje game osv. Poängställningen skall presenteras på ett trevligt sätt. Efter avslutad match ska setsiffror samt vinnaren presenteras.

Skriv en pausfunktion, så att vi hinner med att se hela presentationen (så att vi kan stanna till efter x stycken bollar, game osv.) När användaren ska peka ut en spelare (vilka som ska spela), är det praktiskt att identifiera dem med placeringsnumret. Då slipper användaren mata in hela namnet ...

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

När tennismatcher visas på TV brukar en resultattabell, som ändras efter varje boll och set, visas längst ner i bild. Låt ditt program skriva ut och uppdatera en sådan tabell i ett grafikfönster (istället för att skriva ut resultaten rad för rad).

EECS, KTH
DD100N prgw hösten 2018 (Python)

145 Varuprisdatabas

P-uppgiften ska göras individuellt. Läs EECS: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 streckkodsläsare. 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 streckkodsläsare, 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	502000.00	1004000.00
STÖVLAR	1	159.90	159.90
CHIPS	1	14.90	14.90
Total	5		1004190.00

Ditt program ska uppföra sig på motsvarande sätt. Databasen ska ligga i en varufil, som du skriver ihop själv. Då vi saknar streckkodsläsare matar vi in varorna via kassaapparatens tangentbord. Inmatningen av det ovanstående inköpet får då följande utseende:

```
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	502000.00	1004000.00
STÖVLAR	1	159.90	159.90
Total	5		1004190.00

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 (som i riktiga butiker) 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.

EECS, KTH
DD100N prgw hösten 2018 (Python)

155 Tittarsiffror

P-uppgiften ska göras individuellt. Läs EECS: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.

EECS, KTH
DD100N prgw hösten 2018 (Python)

166 Telefonregister

P-uppgiften ska göras individuellt. Läs EECS: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.

EECS, KTH
DD100N prgw hösten 2018 (Python)

168 Minröjning

P-uppgiften ska göras individuellt. Läs EECS: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.

EECS, KTH
DD100N prgw hösten 2018 (Python)

178 Periodiska systemet

P-uppgiften ska göras individuellt. Läs EECS: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`

EECS, KTH
DD100N prgw hösten 2018 (Python)

184 Patiens med annorlunda kortlek

P-uppgiften ska göras individuellt. Läs EECS: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.

EECS, KTH
DD100N prgw hösten 2018 (Python)

192 Kalender

P-uppgiften ska göras individuellt. Läs EECS: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 filofax. Varje "sida" är en textrad med en minnesanteckning av typen
"`2018-03-17: Träffa de nya deltagarna inför avsnitt 137`"
eller
"`2018-04-01: 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 filofaxen 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: Inför klockslag (starttid och sluttid) för aktiviteter. Nu ska det gå att ha flera aktiviteter per dag, och man ska kunna ändra och ta bort enskilda aktiviteter från en viss dag.

Varna om användaren försöker lägga in en aktivitet som överlappar med en annan.

Man ska också kunna se hela månadens aktiviteter.

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

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

EECS, KTH
DD100N prgw hösten 2018 (Python)

193 Memory

P-uppgiften ska göras individuellt. Läs EECS: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 n ord från filen (18 i exemplet ovan), 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.

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.

EECS, KTH
DD100N prgw hösten 2018 (Python)

195 Zoo

P-uppgiften ska göras individuellt. Läs EECS: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: Gör ett kalendarium för hela året - en fil för varje dag - med information om vad som händer på Zoo just den dagen.

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`

EECS, KTH
DD100N prgw hösten 2018 (Python)

198 Salladsbaren

P-uppgiften ska göras individuellt. Läs EECS: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:

- En person ska ange vad de vill ha i sin sallad.
- Om det finns någon sallad som stämmer överens med de valda ingredienserna så ska alla matchande alternativ ges till användaren.
- Ifall 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 so kan komma ifråga ska den billigaste väljas.)
- Efter att personen har valt sin sallad så ska man komma till menyn för extra val där alla ingredienser och deras priser listas.
- När personen är nöjd så ska ett kvitto skrivas ut på fil.

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.

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.

EECS, KTH
DD100N prgw hösten 2018 (Python)

116 Djurparken

P-uppgiften ska göras individuellt. Läs EECS: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.

EECS, KTH
DD100N prgw hösten 2018 (Python)

111 När vi har tid

P-uppgiften ska göras individuellt. Läs EECS: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.

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.

EECS, KTH
DD100N prgw hösten 2018 (Python)

120 Parkeringshuset

P-uppgiften ska göras individuellt. Läs EECS: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 var 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.

Kunder ska ges möjlighet att få veta hur stor deras skuld för en bil till parkeringshuset är. Parkeringshuset har valt att avrunda upp alla parkeringstider till närmast halvtimme. 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  Inpassage/Utpassage...
   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: Utöka menyn med alternativet att låta kunden betala sin skuld. Skulden ska beräknas ur loggfilen (med in- och utpasseringar) som lästs in. Om kunden betalar mer, så är det pengar till godo. Eftersom parkeringsbolaget ingått hemligt avtal med dataövervakningsmyndigheten får ingen data raderas ur loggfilen.

Extrauppgift, betyg A: Lägg till ett grafiskt gränssnitt.

EECS, KTH
DD100N prgw hösten 2018 (Python)

124 Packlistor för resor

P-uppgiften ska göras individuellt. Läs EECS: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/Två burkar 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/En liten elfläkt(batteridriven)/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: Utöka funktionaliteten till 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!

Vid sökning efter listor ska även datumet tas i beaktande. Om flera packlistor gäller samma dag, skall på samma sätt som tidigare en väljas ut och presenteras.

Extrauppgift, betyg A: Lägg till ett grafiskt gränssnitt.