Regler

- Student får lämna salen tidigast en timme efter tentans start.
- Vid toalettbesök eller rökpaus ska pauslista utanför salen fyllas i.
- All form av kontakt mellan studenter under tentans gång är strängt förbjuden.
- Böcker och anteckningssidor kan komma att granskas av tentavakt i samband med tentans start samt under tentans gång.
- Frågor om specifika uppgifter eller om tentan i stort ska ställas via tentasystemet.
- Systemfrågor kan ställas till assistent i sal.
- Ingen uppgift rättas efter tentatidens slut.
- Ingen kompletteringsmöjlighet ges de sista tio minuterna.
- Så länge en uppgift inte har betyget U kan den kompletteras till godkänt.
- Inga elektroniska hjälpmedel får medtas. Mobiltelefon ska vara avstängd och ligga i jacka eller väska.
- Inga ytterkläder eller väskor får förvaras vid skrivplatsen.

Antal uppgifter	5
Antal sidor (exklusive denna)	5
Hjälpmedel	En pythonbok (tex Learning Python 4 ed.) Ett A4-ark med egna anteckningar

VÄND!

Betygssättning

Tentan innehåller fem uppgifter. För godkänt betyg krävs minst två avklarade uppgifter. För gränser för högre betyg, se i tabell 1.

Tid (timmar)	Antal uppgifter	Betyg
3	3	5
4	4	5
4	3	4

Tabell 1: Betygssättning

Bonus från labserien

Bonus från labserien gäller endast vid förstagångstentamen och går därför inte att använda vid detta tillfälle.

Information

Inloggning

Logga in på ditt normala IDA-konto men välj session “exam system”.

Följ menyvalen så långt det går tills du ska mata in ett engångslösenord. Tag fram ditt LiU-kort och visa det för tentavakten för att få detta lösenord.

Avslutning

Efter att du loggat ut som vanligt ska du vänta ett tag och sedan trycka på knappen “Avsluta tentamen” när det är möjligt. När detta är gjort är det omöjligt att logga in igen.

Uppgift 1

När man jobbar i en kassa och ska ge växel till en kund gör man oftast det i så stora valörer som möjligt (det vore jobbigt att alltid få all växel i enkronor!). Vissa har dock lite svårt för att räkna ut hur mycket växel man ska ge tillbaka och därför är din uppgift att skapa ett program som kan hjälpa.

Ditt program startas med två kommandoradsargument, det första är ett filnamn och det andra ett heltal. På filen som anges ska det finnas en eller flera rader med kommaseparerade heltal som anger de valörer som finns i kassan. Detta för att göra programmet lite framtidssäkrat (du vet väl att vi får nya sedlar i oktober?). Heltalet som anges på kommandoraden motsvarar totalsumman växel som ska ges tillbaka till kunden. Ditt program ska sedan skriva ut en tabell enligt formatet i körexemplen nedan med hur många av varje valör som krävs för att få den angivna totalsumman. Du kan anta att valörerna i den angivna filen alltid är i sjunkande ordning.

Körexempel 1

```
$ python3 change.py given_files/values_now.txt 1223
Du behöver följande:
Valör  Antal
 1000      1
  100      2
   20      1
    1      3
```

Körexempel 2

```
$ python3 change.py given_files/values_oct.txt 1223
Du behöver följande:
Valör  Antal
 1000      1
  200      1
   20      1
    1      3
```

Körexempel 3

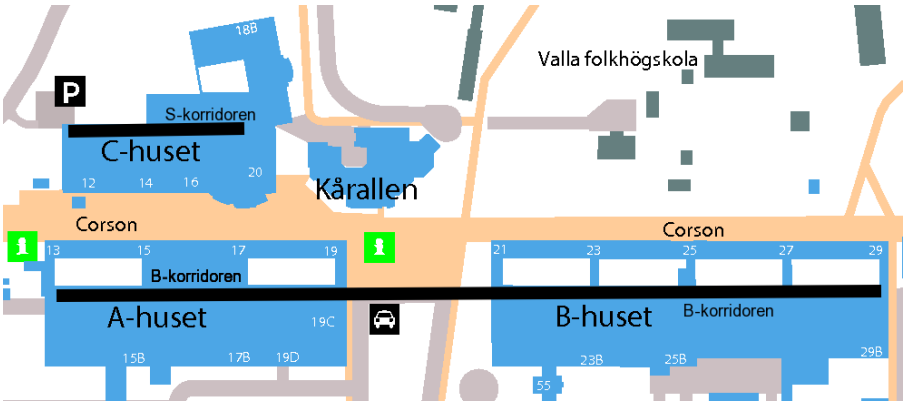
```
$ python3 change.py given_files/values_2016.txt 1223
Du behöver följande:
Valör  Antal
 1000      1
  200      1
   20      1
    2      1
    1      1
```

Uppgift 2

De äldre husen på camups Valla (A,B,C) har ett väldigt logiskt sätt att namnge sina salar. Alla husen är uppbyggda med korridorer som går i ett rutmönster. En korridor går antingen i nord-sydlig riktning och följer därmed Corson (cykelvägen som går genom campus) och har då en bokstav som namn eller så går korridoren i öst-västlig riktning och har då ett nummer som namn. Alla rum har sitt unika nummer på formatet VK:Rum, där V är våning (2 är entréplan), K korridorsnamn och Rum är ett unikt nummer i den korridoren. A- och B-husen har korridorerna A-D, där rumsnummer 400-798 är i B-huset och alla över 800 är i A-huset medan C-huset har korridorerna P-U. Det finns fyra undantag till denna regel, sal C1-C4 har rumsnummer 2C:1 - 2C:4. Alla jämna korridorsnummer är i C-huset (korridor 12-20), alla udda korriderer i intervallet 13-19 är i A-huset medan B-huset har udda korridorer 21-29. I tabell 2 finns några exempel på rum och figur 1 visar en karta med korridor B och S markerade. Vissa rum (som t.ex. föreläsningssalar och konferensrum) har även ett namn som till exempel sal A2 har rumsnummer 2B:908.

Rumsnummer	Plats
3D:450	Våning 3, korridor D, hus B
213:123	Våning 2, korridor 13, hus A
2P:34	Våning 2, korridor P, hus C
3A:912	Våning 3, korridor A, hus A

Tabell 2: Några exempel på rumsnummer



Figur 1: Karta över LiU

På filen *given_files/liu_rooms.py* finns ett program som skapar och hanterar ett antal rum. Din uppgift är att skapa en ADT för att hantera ett rum på modulen *room.py*. Din ADT måste stödja de primitiver som krävs av programmet på filen liu_rooms.py. Du får skapa fler funktioner vid behov för intern användning. Kommentarer i filen liu_rooms.py beskriver något av det som måste uppfyllas av din ADT såsom format på utskrifter och liknande. Du får inte gör några ändringar i den givna filen.

Uppgift 3

Algoritmen *Counting Sort* är en sorteringsalgoritm som är effektiv i tid men kan ha stor minnesanvänding om man har otur (dvs stor spridning på indatats värdemängd). Nedan beskrivs algoritmen i pseudokod:

```
1. Indata: Lista kallad L med icke-negativa värden
2. Skapa en tom lista R
3. För varje värde V i L:
   3.1 Om R har en storlek < V:
      3.1.1 Utöka R med nollor tills den har längd = V+1
   3.2 Öka värdet på index V i R med 1
4. Skapa en ny lista U
5. För varje värde V med index I i R:
   5.1 Sätt in V stycken I i U
6. Returnera U
```

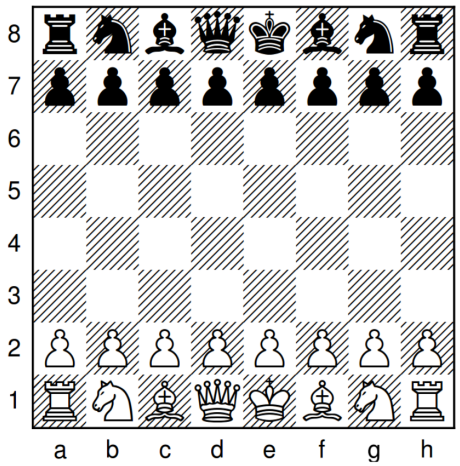
Implementera algoritmen enligt beskrivningen ovan som en fristående funktion och skapa ett program som ber användaren mata in en rad med heltal och skriver ut talen sorterade med hjälp av din funktion enligt körexemplet nedan.

Körexempel 1

```
Mata in heltal: 3 5 2 6 2 6 9 0 19 22 5 23
Talen i sorterad ordning: 0 2 2 3 5 5 6 6 9 19 22 23
```

Uppgift 4

I denna uppgift ska du skapa ett program som ber användaren mata in schackdrag och därefter rita ut spelplanen som den ser ut efter det sista draget är gjort. Varje drag matas in på en rad enligt formatet "startkoordinat"-"slutkoordinat" och inmatningen kan anses vara avslutad när en tom rad matas in. Varje ruta på en schackplan indexeras med en kombination av en bokstav och en siffra enligt figuren nedan som visar startuppställningen. Exempelvis har vit drottning koordinat d1. I tabellen till höger nedan ser du unicode-namnen på de olika figurerna (de kanske ser lite annorlunda ut i din terminal).



Symbol	Namn	Startposition
♔	WHITE CHESS KING	e1
♕	WHITE CHESS QUEEN	d1
♖	WHITE CHESS ROOK	a1 & h1
♗	WHITE CHESS BISHOP	c1 & f1
♘	WHITE CHESS KNIGHT	b1 & g1
♙	WHITE CHESS PAWN	alla i rad 2
♚	BLACK CHESS KING	e8
♛	BLACK CHESS QUEEN	d8
♜	BLACK CHESS ROOK	a8 & h8
♝	BLACK CHESS BISHOP	c8 & f8
♞	BLACK CHESS KNIGHT	b8 & g8
♟	BLACK CHESS PAWN	alla i rad 7

Körexempel 1

```
Mata in drag:
e2-e4
e7-e5
f1-c4
f8-c5
d1-h5
c5-f2
h5-f7
```

♜♞♝♛♚ ♞♜
♟♟♟♟ ♕♟♟

  ♟
♗ ♙

♙♙♙♙ ♝♙♙
♖♘♗ ♔ ♘♖

TIPS: I Python kan man använda litteralen `'\N{BLACK CHESS PAWN}'` för att få fram tecknet ♟
TIPS 2: Tänk på att terminalens bakgrund är svart så färgerna kan se fel ut...

Uppgift 5

Skriv ett program som ber användaren mata in ett heltal. Programmet skapar sedan en lista med så många heltal med slumpade värden i intervallet $[0, 59]$. Ditt program ska skriva ut dessa tal på skärmen i rader med 10 tal per rad. Därefter ska ditt program skriva ut alla godkända tidsstämplar på formatet hh:mm (timme:minut) man får när man parvis slår ihop två närliggande tal i listan. Observera att vissa av talen kan ingå i två av tidsstämplarna.

Körexempel 1 (Användarinmatning är i fet stil)

```
Mata in ett heltal: 20

20 slumpade tal:
01 42 23 59 56 32 14 25 59 09
41 03 18 53 38 12 24 15 27 32

Korrekta tider:
01:42
23:59
14:25
09:41
03:18
18:53
12:24
15:27
```

Körexempel 2

```
Mata in ett heltal: 19

19 slumpade tal:
39 56 06 59 08 14 47 41 32 56
25 06 33 20 17 14 44 01 45

Korrekta tider:
06:59
08:14
14:47
06:33
20:17
17:14
14:44
01:45
```