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.) En A4-sida 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. Om du har, av LiU godkänd, förlängd skrivtid gäller istället tabell 2.

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

Tabell 1: Betygssättning

Tid (timmar)	Antal uppgifter	Betyg
4	3	5
5.5	4	5
5.5	3	4

Tabell 2: Betygssättning vid förlängd skrivtid

Bonus från labserien

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

Information

Inloggning

Logga in på tentakontot med följande användaruppgifter:

```
Användarnamn: examx
Lösenord:     kluring1
```

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.

När du är inloggad är det viktigt att du startar tentaklienten genom att högerklicka på bakgrunden och välja “tentaklient” i menyn.

Avslutning

Tryck på knappen märkt “exit” i menyn längst nere på skärmen och välj ok. Vänta ett tag och tryck sedan på knappen “Avsluta tentamen” när det är möjligt. När detta är gjort är det omöjligt att logga in igen.

Uppgift 1

Sture har skapat ett litet testprogram som heter `given_files/config_program.py` för att hantera konfigurationsfiler. Programmet använder sig av en abstrakt datatyp som hanterar en fil med konfigurationsdata. Tyvärr har modulen som innehåller denna ADT försvunnit och din uppgift är att rekonstruera denna utifrån de anrop som används i Stures program.

Konfigurationsfiler kan innehålla två typer av data, heltal och textuella värden. Alla rader har formatet `Nyckel = Värde` (mellanslag viktigt). Det finns även rader som inleds med `#`, dessa är kommentarsrader och ska ignoreras av din ADT. Endast strängar kan användas som nycklar och din ADT ska göra skillnad på versaler och gemener, dvs nyckeln "Hej" och "hej" är olika. Om en nyckel förekommer flera gånger i konfigurationsfilen ska endast det senaste värdet användas. Se den givna filen `given_files/testfile.cf` för exempel på konfigurationsdata.

Din modul måste åtminstone innehålla de funktioner som används i Stures program, men du får även skapa andra funktioner vid behov.

I körexemplet nedan körs programmet med den givna konfigurationsfilen. Det är dock inte nödvändigt att nycklarna skrivs ut i samma ordning (dock ska de såklart ha samma värden).

Körexempel 1

```
name => Sture
x => hej
X => 10
text => En liten text ...
Text => En annan text

10*4=40
```

KRAV: Du får inte göra några ändringar i det givna programmet! Endast din skapade modul behöver skickas in.

Uppgift 2

Syföreningen "Astas Tanter" brukar träffas ofta och ha trevligt tillsammans. För att slippa hålla reda på skulder som t.ex. att "Asta är skyldig Greta 15 kr för en kaffe" har gruppen kommit på ett skuldsystem. När man köper något åt någon annan i föreningen skriver man upp transaktionen med ett positivt tal för sig själv och ett negativt tal för den man handlat åt. Detta leder till att den som har en negativ totalsumma är skyldig systemet pengar och bör lösa det genom att köpa något åt någon som har en positiv totalsumma.

Din uppgift är att skriva ett program som hanterar syföreningens skuldsystem. Programmet ska låta användaren mata in transaktioner på formatet `Namn: värde`, där `värde` är ett positivt eller negativt heltal (alla inköp avrundas till hela kronor). Om användaren matar in texten `skriv ut` ska en sammanställning skrivas ut sorterad efter totalsumma enligt körexemplet nedan. Programmet ska köra vidare tills det avbryts med Ctrl-C

Körexempel 1

```
Välkommen till Astas Tanters skuldlista!
Mata in transaktioner:
Asta: -32
Inga: -54
Greta: +86
Asta: +23
Stina: -23
skriv ut
--------------------
Utskrift av skulder:
Inga:            -54
Stina:           -23
Asta:             -9
Greta:           +86
--------------------
Mata in transaktioner:
Greta: -25
Asta: +25
skriv ut
--------------------
Utskrift av skulder:
Inga:            -54
Stina:           -23
Asta:            +16
Greta:           +61
--------------------
Mata in transaktioner:
```

Uppgift 3

Har du någon gång spelat spelet MSRöj? (Minesweeper) Det är ett spel som följer med ett populärt operativsystem som går ut på att markera alla rutor i ett rutnät som innehåller minor. Klickar man på en ruta som innehåller en mina har man förlorat spelet. Klickar man på någon annan ruta får man normalt sett veta hur många minor som finns i närheten av den rutan. I denna uppgift kommer användaren mata in spelplanen med asterisker (*) för att markera minor och punkter (.) för att markera säkra positioner. Inmatningen inleds med två tal, antal rader och kolumner.
Din uppgift är att skriva ut samma spelplan på skärmen men med alla punkter ersatta med heltal som motsvarar antal närliggande minor. En närliggande mina är en mina som ligger i en 3x3-omgivning aktuell position.

Du kan anta att användaren endast matar in en korrekt spelplan, dvs alla rader är lika långa och innehåller endast punkter och asterisker.

Körexempel 1

```
Mata in spelplanen:
4 4
*...
....
.*..
....

Resultat:
*100
2210
1*10
1110
```

Körexempel 2

```
Mata in spelplanen:
3 5
**...
.....
.*...

Resultat:
**100
33200
1*100
```

Uppgift 4

Ett palindromtal är ett heltal som även är ett palindrom (blir samma om man läser det framifrån och bakifrån). Din uppgift är att skriva ett program som generar palindromtal enligt följande algoritm:

1. Utgå ifrån ett tal N
2. Avbryt om N är ett palindromtal
3. Summera talet N med samma tal läst bakifrån
4. Börja om från steg 2 med summan från föregående steg som N

Räknexempel:

```
N=195

 195 +  591 =  786 (ej palindrom)
 786 +  687 = 1473 (ej palindrom)
1473 + 3741 = 5214 (ej palindrom)
5214 + 4125 = 9339 (palindrom, klar)
```

Ditt program ska be användaren mata in talet N och skriva ut dels hur många iterationer som krävdes samt det resulterande palindromtalet, enligt körexemplen nedan. Programmet ska be om ny inmatning tills användaren matar in tal 0. Det är inte säkert att alogritmen fungerar för alla tal. Därför ska sökningen avbrytas efter 50 iterationer om inget palindrom hittats.

Körexempel 1

```
Mata in N: 195
Palindrom 9339 nåddes efter 4 iterationer.
Mata in N: 265
Palindrom 45254 nåddes efter 5 iterationer.
Mata in N: 750
Palindrom 6666 nåddes efter 3 iterationer.
Mata in N: 2
Palindrom 2 nåddes efter 0 iterationer.
Mata in N: 196
Sökningen avbröts efter 50 iterationer.
Mata in N: 0
Avslutar!
```

Uppgift 5

En stor del av världens datorkraft går åt till att sortera data. Vi ska inte vara sämre så här ska du implementera en sorteringsalgoritm. Skapa en funktion `sort` som implementerar algoritmen i ruta 1. Skapa därefter ett huvudprogram som tar emot tre heltal på kommandoraden, härmed kallade A, B och C. Programmet ska slumpa en lista med C värden i intervallet $[A, B]$, skriver ut listan, sorterar den och skriver ut den igen.

Du kan anta att användaren alltid ger korrekta värden på kommandoraden.

```
Indata: en lista med index 0..N
För varje index i, i intervallet 0..N-1, gör följande:
  Sök alla index i+1..N efter det minsta värdet
  Kalla det index som har minsta värdet för i_min.
  Om värdet på index i_min är mindre än värdet på index i:
    byt plats på index i och i_min
Returvärde: Inga (ändrar på indatat)
```

Ruta 1: Sorteringsalgortim

Körexempel 1

```
zaza10: my_sort.py 10 15 14
[11, 14, 12, 14, 13, 12, 11, 15, 14, 10, 12, 10, 11, 13]
[10, 10, 11, 11, 11, 12, 12, 12, 13, 13, 14, 14, 14, 15]
```