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

Varje avklarad deadline ger fem minuter extra till gränserna för högre betyg.

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

Kommandot `paste` tar, som ni alla vet, två filer som argument. Därefter skriver det ut filernas innehåll brevid varandra radvis separerat med ett tabulatortecken (`\t`). Programmet kan även ta emot väljaren `-d` följt av ett tecken som ska användas som separator mellan de båda filerna istället för `\t`. I denna uppgift ska ni göra ett program som fungerar på samma sätt som `paste` men separerar med mellanslag som standard.

Antag att vi har två filer, `A.TXT` och `B.TXT` med följande innehåll:

A.TXT:

```
Detta är en

liten fil
en
```

B.TXT:

```
längre text
som är större än en

tom rad kan

ställa till det.
```

Körexempel 1

```
zaza1$ paste.py A.TXT B.TXT
Detta är en längre text
 som är större än en
liten fil
en tom rad kan

 ställa till det.
```

Körexempel 2

```
zaza1$ paste.py A.TXT B.TXT -d|
Detta är en|längre text
|som är större än en
liten fil|
en|tom rad kan
|
|ställa till det.
```

TIPS: Det kan finnas tomma filer...
KRAV: Du får inte spara undan hela filernas innehåll utan ska hantera dem radvis.

Uppgift 2

Skriv en funktion ***merge*** på filen ***merge.py*** som slår ihop två listor som vi härmed kallar A och B till en ny lista med namn C som returneras av funktionen. Det är givet att en av listorna A och B alltid är längre än den andra. När ni skapar lista C ska elementen i den kortare listan av A och B (som vi kallar K) separeras så jämnt som möjligt bland elementen i den längre (kallas L). Om det inte blir jämnt avstånd mellan elementen ska "överblivna" element från L läggas till eller tas bort från slutet av C.

Körexempel 1

```
>>> A = [1, 2, 3, 4]
>>> B = ['a', 'b', 'c']
>>> merge(A, B)
[1, 'a', 2, 'b', 3, 'c', 4]
```

Körexempel 2

```
>>> A = [1, 2, 3, 4, 5]
>>> B = ['a', 'b']
>>> merge(A, B)
[1, 2, 'a', 3, 4, 'b', 5]
```

Körexempel 3

```
>>> A = ['a', 'b', 'c']
>>> B = [1, 2, 3, 4, 5]
>>> merge(A, B)
[1, 'a', 2, 'b', 3, 'c', 4, 5]
```

Körexempel 4

```
>>> A = [1, 2, 3, 4, 5, 6]
>>> B = ['a', 'b']
>>> merge(A, B)
[1, 2, 'a', 3, 4, 'b', 5, 6]
```

Denna algoritm kanske inte är den bästa, se t.ex. vad som händer med följande listor:

```
>>> A = list(range(1,9))
>>> B = ['A', 'B', 'C', 'D', 'E']
>>> merge(A, B)
[1, 'A', 2, 'B', 3, 'C', 4, 'D', 5, 'E', 6, 7, 8]
```

Uppgift 3

På filen ***given_files/BINARY.TXT*** finns det flera rader. Varje rad representerar ett heltal skrivet i binär talbas (talbas 2). Din uppgift är att skriva ett program som för varje tal på filen skriver ut den hexadecimala formen av samma tal på skärmen.

Antag att filen innehåller följande:

```
1001010
10110
001011011010010
```

Då ska följande bli utskriften av ditt program:

```
4A
15
16D2
```

För er som inte kommer ihåg hur man omvandlar ett tal från binär talbas till hexadecimalt kommer här en kort förklaring:

1. Fyll ut talet med nollor i början så att talets längd blir en jämn multipel av fyra
2. Tag fyra siffror ur talet och kalla detta för A
3. Tag reda på As decimala värde (talbas 10) genom att summera de olika delarna i talet. Ett fyrasiffrigt binärt tal har 8-talssiffra, 4-talssiffra, 2-talssiffra och entalssiffra. Talet 1001_2 har då värdet $1 * 2^3 + 0 * 2^2 + 0 * 2^1 + 1 * 2^0 = 9_{10}$
4. Om värdet i föregående steg är under 10 har det samma värde i hexadecimal talbas, annars används bokstäverna A-F för att symbolisera värdena 10-15 ($A = 10, B = 11, \ldots, F = 15$).
5. Upprepa från steg 2 (med nästa fyra siffror) för hela talet.

Uppgift 4

Skalärprodukten av två vektorer $v_1 = (x_1, x_2, \ldots, x_n)$ och $v_2 = (y_1, y_2, \ldots, y_n)$ beräknas enligt $v_1 \cdot v_2 = x_1 * y_1 + x_2 * y_2 + \cdots + x_n * y_n$. Skriv ett program som ber användaren mata in två vektorer och därefter beräknar den minsta skalärprodukten som är möjlig att få med talen i de två vektorerna. Ditt program ska alltså permutera (flytta elementen) vektorerna för att få den minsta skalärprodukten. Det är givet att användaren matar in lika långa vektorer. Det är produkten som är viktig, om flera permutationer ger samma resultat spelar det ingen roll vilken av dem som skrivs ut.

Körexempel 1

```
Mata in första vektorn:  1 2 3 4 5
Mata in andra vektorn: 1 0 1 0 1
Minsta skalärprodukten: (1 4 2 5 3)*(1 0 1 0 1)=6
```

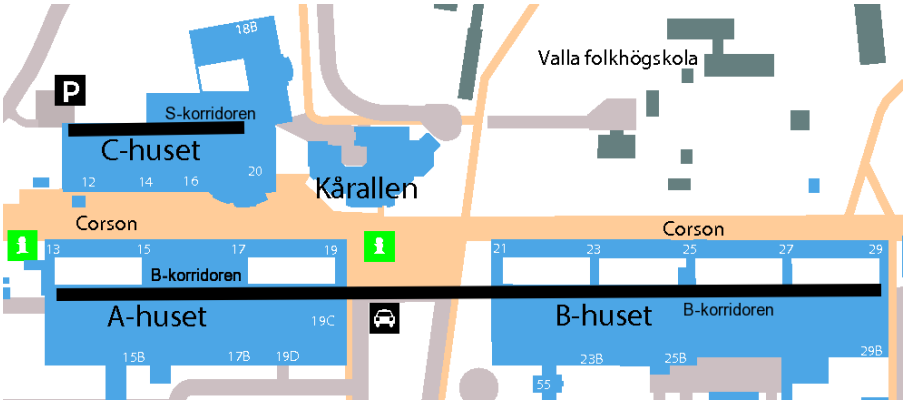
TIPS 1: Det räcker att ändra ordningen i den ena av vektorerna.
TIPS 2: Modulen `itertools` har en funktion `permutations` för att generera permutationer.

Uppgift 5

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 3 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 3: 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.