Regler

- Student får lämna salen tidigast en timme efter tentans start.
- Vid toalettbesök 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 tabell 1.

Tid (timmar)	Antal uppgifter	Betyg
3	3	5
4	4	5
4	3	4
5	2	3

Tabell 1: Betygssättning

Bonus från labserien

Vid omtentamen ges ingen bonus från labserie.

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.

Tentan

Tentan kommer publiceras elektroniskt i form av ett pdf-dokument med namn `exam.pdf` i katalogen `given_files` när tentan startar. Observera att katalogens innehåll inte är läsbar innan tentans start.

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

En simhall har flera banor som alla kan bokas av diverse grupper. I denna uppgift ska du, i modulen `booking_calendar.py`, skapa en ADT för att representera en bokningskalender för dessa banor. En bana har ett bannummer (heltal från 1 till antal banor i bassängen) och kan bokas på minutnivå. I mappen `given_files` finns en modul `my_time.py` som du ska använda dig av för att representera klockslag, både i gränssnittet och internt i din ADT. Åtminstone följande primitiver ska stödjas av din ADT:

`create_calendar(num)` Returnerar en bokningskalender för `num` antal banor.

`reserve(calendar, num, start, stop)` Lägg till bokning i kalendern `calendar` för bana `num` med starttid `start` och sluttid `stop`. Returnerar `True` om bokningen lyckades och `False` om banan är felaktig eller om bokningen överlappar en redan existerande bokning för banan.

`available(calendar, num, time)` Returnerar `True` om banan `num` är ledig tiden `time` i kalendern `calendar`, annars `False`.

`available_lanes(calendar, time)` returnerar hur många banor som är lediga tiden `time` i kalendern `calendar`.

Du får lägga till andra primitiver om du anser det nödvändigt. Dessutom förväntas ett litet testprogram där du testar primitivernas funktionalitet på ett bra sätt. Vilken intern representation du väljer är helt upp till dig.

Uppgift 2

Sam har hittat ett nytt sätt att dela ut veckopeng till sina barn. Varje dag kan maximalt ett barn få pengar och varje dag motsvarar en krona. När ett barn fått en utbetalning får barnet välja hur många dagar det vill vänta tills nästa betalning (och därmed bestämmer det hur många kronor det får). Din uppgift är att simulera en period av utbetlningar enligt följande algoritm:

1. Användaren matar in barnens namn (ett ord, varje bokstav representerar ett barns namn)
2. Det första barnet får en krona dag 1, det andra barnet två kronor dag 2 osv för varje barn
3. Det första barnet (i tidsordning) får sin utbetalning och får därefter välja antal dagar tills nästa utbetalning. Om den dagen redan har en betalning väljs automatiskt nästa lediga dag. Detta upprepas för alla valda dagar som matas matas in på en rad.
4. När programmet är klart skrivs en rad text ut. En dag utan utbetalning representeras av ett bindestreck och vid en dag med betalning visas barnets bokstav.

Körexempel 1

```
Mata in barnens namn: ABC
Mata in barnens valda dagar: 3 2 1
Utbetalningar: ABCABC
```

Körexempel 2

```
Mata in barnens namn: ABC
Mata in barnens valda dagar: 1 2 3
Utbetalningar: ABCABC
```

Körexempel 3

```
Mata in barnens namn: ABC
Mata in barnens valda dagar: 5 10 3 2
Utbetalningar: ABC--ACA---B
```

De två första exemplen gav samma resultat eftersom det blev krockar. I exempel 2 kunde inte barn A vänta en dag eftersom B redan fått den dagen (och inte heller två dagar eftersom C har den) och fick därför vänta till fjärde dagen till nästa betalning.

Uppgift 3

I denna uppgift ska du skapa ett program som låter användaren mata in en sträcka i centimeter och därefter slupa en enhet för att slutligen skriva ut vad sträckan motsvarar i den nya enheten. Nedan följer de enheter ditt program ska stödja samt hur du omvandlar till dem. Ditt program ska skriva ut resultatet avrundat till ett heltal.

Enhet	Motsvarar (i cm)
tum	2,54
fot	30,48
aln	59,3808
yard	91,44
meter	100
mile	160934
millimeter	0,1

Tabell 2: Omvandligstabell mellan enheter som ska stödjas

Körexempel 1

```
Mata in en sträcka i cm: 38654
Detta är 432 yards.
```

Körexempel 2

```
Mata in en sträcka i cm: 38
Detta är 380 millimeter.
```

Körexempel 3

```
Mata in en sträcka i cm: 412
Detta är 14 fot.
```

Uppgift 4

Givet en textfil med ett antal heltal separarerade med ett antal vita tecken ska du i detta problem kontrollera om det finns någon dubblett och sedan skriva ut dubbletten. Det är givet att det finns maximalt en dubblett i filen. Filens namn ges av användaren på kommandoraden. Om ingen fil anges ska programmet skriva ut ett felmeddelande. Ditt program ska inte läsa in fler rader av filen än vad som behövs för att lösa uppgiften.

Körexempel 1

```
$ python3 duplicates.py given_files/has_duplicate.txt
Siffran 19 var en dubblett.
```

Körexempel 2

```
$ python3 duplicates.py given_files/no_duplicate.txt
Filen innehöll ingen dubblett.
```

Körexempel 3

```
$ python3 duplicates.py
Ingen fil angavs!
```

Uppgift 5

I uppgift 4 krävdes det filer med heltal separerade med en kombination av whitespace. I denna uppgift ska du skapa ett program för att skapa sådana filer. Programmet börjar med att be användaren mata in namnet på filen som ska skapas samt hur många tal som ska genereras. Dessutom ska användaren tillfrågas om det ska finnas en dubeltt och isåfall vilket värde som ska dupliceras. Nedan följer en lista med krav på den genererade filen:

1. Talen som genereras är i intervallet $[1, 10000]$
2. Mellan varje tal följer ett till tio vita tecken (slumpad kombination av tecknen `' '`, `'\t'` och `'\n'`)
3. Antal tal ska stämma exakt med det användaren matade in. Om dubblett efterfrågas ska exakt en dubblett av det inmatade talet finnas i filen (dvs det ska finnas på två ställen)

Körexempel 1

```
Mata in filnamn: duplicate.txt
Hur många värden ska genereras: 500
Ska det finnas en dubblett [j/n]: j
Vilket värde ska dubbleras: 19
Filen "duplicate.txt" har genererats.
```

Körexempel 2

```
Mata in filnamn: no_duplicate.txt
Hur många värden ska genereras: 200
Ska det finnas en dubblett [j/n]: n
Filen "no_duplicate.txt" har genererats.
```