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 extra bonus från labserien.

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

I grundskolan fick du lära dig att ställa upp tal när man skulle summera dem. I denna uppgift ska du skapa ett program som kontrollerar om summeringen gått rätt till.
Först kommer användaren mata in ett antal tal. Varje tal matas in på en egen rad och varje siffra i talet separeras med minst ett mellanslag. När samtliga tal matats in kommer en rad som endast innehåller bindestreck. Till slut kommer summan (även den med siffror separerade med mellanslag).
Din uppgift är alltså att komtrollera att summan av alla tal som matades in är samma som summan som matades in.

Körexempel 1

```
Mata in talen:
1 1 2 3
  1
  2 0 0
-------
1 4 2 3

Summan är felaktig
```

Körexempel 2

```
Mata in talen:
  1
2 0
-
2 1
Summan är korrekt
```

Körexempel 3

```
mata in talen:
 1 1
-1 2
--
 2

Summan är felaktig
```

Uppgift 2

Skriv ett program som läser in ett positivt heltal `N`. Programmet skall därefter läsa in 3 ”operationer” som skall ”köras” (i ordningen man skrev dem) på talet N. Operationerna kan antingen vara ”DUP” eller ”ROT”.

En DUP-operation innebär att man längst till höger i N lägger till en dubblett av siffran längst till höger. Om man t.ex. kör DUP på 134 så blir det 1344.

En ROT-operation innebär att man flyttar siffran som är längst till höger i N längst till vänster. Om man t.ex. kör ROT på 1324 så blir det 4132.

Operationerna matas in på egna rader. Programmet skall slutligen skriva ut det modifierade N-värdet.

Körexempel 1

```
Mata in N: 134
Mata in 3 operationer:
DUP
DUP
ROT
N blev 41344.
```

Körexempel 2

```
Mata in N: 12345
Mata in 3 operationer:
ROT
ROT
ROT
N blev 34512.
```

Körexempel 3

```
Mata in N: 21
Mata in 3 operationer:
ROT
DUP
DUP
N blev 1222.
```

Uppgift 3

Om du skapar en kvadrat med sidan s (enheten är inte viktig) och i den har en perfekt cirkel med radien r kan du räkna ut ett närmevärde på Pi med hjälp av slumptal. Detta går till på följande sätt: Du slumpar ut en punkt (två koordinater) i kvadraten. Om denna punkt har ett avstånd från centrum som är mindre än eller lika med cirkelns radie så ingår denna punkt i cirkelns area. Alla punkter som slumpas ingår i kvadratens area. Se figur 1 för exempel.

(a) En punkt slumpad i cirkeln (b) En punkt utanför cirkeln

Figur 1: Slumpade punkter i och utanför cirkeln

Genom att hoppas på att punkterna fördelas jämt över kvadratens yta kan vi anse att antalet punkter inom cirkeln borde vara i proportion till cirkelns area och antalet punkter i kvadraten är proportionellt mot kvadratens area. Vi vet formlerna för att räkna ut kvadraten och cirkeln men ser vi pi som en variabel och löser ut den så får vi:

$$\frac{A_{cirkel}}{A_{kvadrat}} = \frac{Pi * r^2}{s^2}$$

$$Pi = \frac{s^2 * A_{cirkel}}{r^2 * A_{kvadrat}}$$

Exakt hur lång sidan på kvadraten är kan du själv välja, men radie=1 och sida=2 kan vara vettiga värden. Ju större kvadrat du har, desto noggrannare borde resultatet bli. Slumpa 1 000 000 000 punkter och för varje 10 000 000:te skall man skriva ut vilket närmevärde man ligger på för tillfället (med 6 decimaler).

Körexempel 1

```
   10000000    3.141577
   20000000    3.141327
   30000000    3.141158
   40000000    3.141176
   50000000    3.141273
   60000000    3.141269
   70000000    3.141227
   80000000    3.141196
   90000000    3.141136
  100000000    3.141144
(programmet fortsätter ett bra tag till...)
```

Uppgift 4

I matematiken finns det ett begrepp som kallas rest vid division. Om man t.ex. delar 7 med 5 så får man 2 i rest, eftersom 5 går i 7 en gång men det blir 2 över. Man kan också säga att resten är det man får när man inte kan dra bort fler nämnare från täljaren. Om vi t.ex. delar 41 med 10 så kan vi dra bort 10 tills vi bara har en 1:a kvar.

Du skall nu göra en rekursiv funktion `calc_rest` som tar två heltalsparametrar och beräknar resten vid division. Funktionen skall returnera resten. Divisionen skall utföras genom upprepad subtraktion. Du får anta att täljaren och nämnaren alltid är större än 0. Funktionen skall också skriva ut varje steg (se körexemplen).

Körexempel 1

```
Ange täljare och nämnare:
7 5
7  5 = 2
Resten blev 2
```

Körexempel 2

```
Ange täljare och nämnare:
41 10
41  10 = 31
31  10 = 21
21  10 = 11
11  10 = 1
Resten blev 1
```

Körexempel 3

```
Ange täljare och nämnare:
44 18
Resten blev 4
```

Uppgift 5

Om man vill vara en häftig hacker så skriver man saker på s.k. leet speak. I denna uppgift skall du göra ett program som översätter från vanlig engelska till detta vackra språk. Här är regler för översättningen:

- Siffror som redan förekommer i text ändras aldrig.
- Förekomster av bokstaven a (både stora och små) byts ut mot tecknet '4'.
- Förekomster av bokstaven e (både stora och små) byts ut mot tecknet '3'.
- Förekomster av bokstaven o (både stora och små) byts ut mot tecknet '0'.
- Förekomsten av bokstaven l (både stora och små) byts ut mot tecknet '1'.
- Förekomsten av bokstaven t (både stora och små) byts ut mot tecknet '7'.
- Övriga bokstäver ska ändras så att det är varannan liten och varannan stor bokstav.
- Andra tecken ska ej modifieras.

KRAV: Du skall skapa ett underprogram `leetspeakify` som tar en sträng som parameter och returnerar den modifierade strängen. T.ex. skall följande program skriva ut "1337 h4Xx0R" på skärmen:

```
S = leetspeakify("leet haxxor")
print(S)
```

Gör ett huvudprogram som tar en text på kommandoraden och skriver ut denna i leet speak.

Körexempel 1

```
$ program.py leet haxor speak
1337 h4X0r Sp34K
```