DUGGA 2017-12-12

Tentaupplägg

TIPS 1:

Läs igenom ALLA uppgifterna. Välj den du känner är lättast först. Det kan gärna ta 10-20 minuter. Försök skriva saker som kan vara problem i uppgifterna. Är det något du absolut kommer att fastna på så kanske det är fel uppgift att ge sig på. Tiden du lägger på att noga läsa uppgifterna tjänar du in på att välja rätt uppgift.

TIPS 2:

Kolla ibland till kommunikationsfönstret. Det kan ha kommit information till alla utan att ni skickat in en fråga. Kanske gäller det dig också (d.v.s. den uppgift du jobbar med).

TIPS 3:

Om ni har problem med kompilator, Emacs eller annat som INTE har med uppgifterna att göra, räck upp handen så kommer en assistent. Detsamma gäller om hur man kopierar givna filer
" cp given_files/* . " eller liknande.

Frågor om själva uppgifterna tar vi i första hand via tentasystemet.

I körexemplen har vi markerat det som användaren matar in på tangentbordet med ***fet och kursiverad*** stil. Tänk på att körexemplen bara är *ett* exempel på när programmet körs. Testa ditt program noga och tänka över hur programmet skall fungera vid andra indata.

Vi hinner normalt sett inte svara på frågor de sista 10 minuterna på tentan. Då ägnar vi all tid åt att rätta uppgifter för att alla skall hinna få svar innan ni går hem.

Betygsgränser:	**Tid**	**TekFak**	**FilFak**
1 poäng	17:00	Betyg 3	Betyg G
2 poäng	16:30	Betyg 4	
2 poäng	15:30		Betyg VG
2 poäng	15:00	Betyg 5	
>=3 poäng	16:30	Betyg 5	
>=3 poäng	16:30		Betyg VG

OBS: Delpoäng delas inte ut på uppgifterna. För att få poäng på en uppgift måste man alltså lösa uppgiften helt (och enligt specifikation).

Lycka till med tenterandet och hoppas att alla får G på minst en uppgift idag.
M.v.h.
/Torbjörn (examinator)

Uppgift 1 - bred eller lång fil? [1p]

Skriv ett program som läser från filen DATA.TXT och avgör om filen är *bred* eller *lång*.

Vi säger att en fils *bredd* är längden på längsta raden. I denna längd räknar vi inte radslutstecknet (enter). Raderna i DATA.TXT kan vara godtyckligt långa.

Vi säger att en fils *längd* är antalet rader i filen. Det kan vara godtyckligt många rader i DATA.TXT.

Programmet skall ange om filen är *lång* (längre än vad den är bred), *bred* (bredare än den är lång) eller inget dera (lika lång som bred).

Vi har olika varianter på DATA.TXT som vi testar med.

Körexempel 1 (DATA.TXT med 70 rader och längsta raden är 64 tecken):

```
DATA.TXT är lång.
```

Körexempel 2 (DATA.TXT med 40 rader och längsta raden är 100 tecken):

```
DATA.TXT är bred.
```

Körexempel 3 (DATA.TXT med 25 rader och längsta raden är 25 tecken):

```
DATA.TXT är varken lång eller bred.
```

Uppgift 2 - Decimaler på vift [1p]

Programmet *Ultra Fast Calculator 2K* (förkortas UFC2K) är helt omatchat i industrin på att göra flyttalsberäkningar i hög hastighet. Problemet är bara att mjukvaruutvecklarna som skapade programmet aldrig lärde sig sin lab 0 särskilt bra och formatteringen på utdatan har blivit helt oläslig! Här är ett par exempel på hur flyttal som programmet matar ut kan se ut:

```
2681589                48128
          -15    3
       0                      182344
```

Dessa tre rader motsvarar talen 2681589.48128, -15.3 och 0.182344.
Mer exakt är formatet följande för varje rad:
[B][H][B][P]

Där [B] är ett godtyckligt antal blanksteg, [H] är ett heltal i intervallet [-999999999,999999999] och [P] är ett heltal i intervallet [1, 999999999][*].

Din uppgift är att skriva ett program som avkodar denna utdata (i detta fall låter vi en människa mata in UFC2Ks utdata direkt på tangentbordet) och som skriver ut varje flyttal (d.v.s. varje rads data), formaterat med plats för 10 tecken i heltalsdelen och 9 decimaler. Användaren fortsätter att mata in flyttal på UFC2Ks format tills programmet avslutas med Ctrl-C.

Körexempel:

```
2681589                48128
   2681589.481280000
          -15    3
       -15.300000000
       0                      182344
         0.182344000
```

TIPS: Använd inte flyttal över huvudtaget, då får du garanterat avrundningsfel. Se heltalsdelen och decimaldelen som två separata heltal.

---

* Observera att eftersom ingen decimaldel som matats ut av UFC2K börjar på 0 så är det mycket troligt att de har en bug i sitt program. Om vi t.ex. tar flyttalet 1.05 så har UFC2K råkat mata ut detta som heltalen 1 och 5. Detta går nu inte att skilja från hur t.ex. 1.5 eller 1.005 skulle matas ut. Detta är dock något som inte du behöver lösa i denna uppgift, eftersom problemet härstammar från ett logiskt fel i UFC2Ks källkod.

Uppgift 3 - Reformationen [2*p]

Du kan göra endast del A för 1 poäng. För 2 poäng krävs både del A och del B.

I år är det precis 500 år sedan Martin Luther spikade upp sina 95 teser på kyrkoporten i Wittenberg, och på så sätt startade reformationen i europa. I ett strategispel simuleras reformationens spridning med tärningsrullande. Två spelare rullar tärningar för varje europeisk stad som håller på att konverteras till den nya tron. Den ena spelaren representerar Luther och de protestantiska krafterna, och den andre representerar påvestaten. Hur många tärningar de två spelarna får beror på många faktorer, t.ex. om Luther finns i staden, hur många angränsande städer som tillhör protestantism/katolicism, och om det finns protestantiska/katolska trupper i närheten. När båda spelarna har slagit sina tärningar så vinner den som fick det högsta slaget. Vid lika går det till protestanterna. Om protestanterna vinner så har staden konverterats till protestantism. Om påvestaten vinner så är staden fortsatt katolsk.

Ett exempel: *Reformationen har kommit till Prag. Protestanterna får 4 tärningar (2 för att Wittenberg redan är protestantiskt och för de protestantiska trupperna där, och 2 för att Luther finns i Prag). Påvestaten får 5 tärningar (3 från de närbelägna katolska städerna Leipzig, Brunn och Linz och 2 för katolska trupper i Prag). Protestanterna slår 2, 3, 3, 4. Katolikerna slår 1, 1, 4, 6, 6. Prag är fortsatt katolskt eftersom katolikerna högsta tärning var högre än protestanternas.*

Del A - [1p]
Skriv en funktion som tar två heltalsparametrar **P** och **K** där P är antalet tärningar som protestanterna får och K är antalet tärningar som katolikerna får. Funktionen skall returnera ett flyttal: sannolikheten (i procent) för att staden blir konverterad till protestantism. Du skall uppskatta sannolikheten genom att simulera förloppet, **gör så här:**

1. Slumpa tal som i exemplet ovan och avgör vilken sida som vann.
2. Utför steg 1 10000 gånger och räkna hur många gånger som protestanterna vann.
3. Dela resultatet med 10000 för att få ut sannolikheten, multiplicera med 100 (för att få det i %).

Del B - [1p]
Givet funktionen du gjorde i del A, skriv ett program som anropar den för olika värden på P och K. Programmet skall skriva ut en tabell över sannolikheten att protestanterna lyckas konvertera städer. Tabellen skall ha 8 rader och kolumner. Kolumnerna motsvarar antalet protestantiska tärningar och raderna antal katolska tärningar.

Körexempel:

```
 PROT:1     2     3     4     5     6     7     8
K 1 58.2% 74.3% 83.4% 87.2% 90.5% 92.7% 94.1% 95.0%
A 2 41.7% 61.6% 71.6% 78.0% 83.2% 87.7% 90.0% 91.4%
T 3 34.1% 53.3% 64.6% 73.0% 78.7% 83.5% 87.0% 89.4%
O 4 29.7% 47.8% 59.1% 68.8% 74.9% 79.1% 83.9% 86.9%
L 5 26.4% 43.6% 55.7% 65.2% 72.1% 77.4% 82.3% 84.8%
S 6 24.9% 41.4% 54.4% 62.4% 70.3% 75.6% 80.3% 83.2%
K 7 22.8% 38.1% 51.3% 59.8% 67.3% 73.4% 78.0% 82.5%
A 8 21.1% 37.1% 49.0% 58.7% 65.8% 72.4% 77.5% 81.5%
```

TIPS: Det finns en given funktion die_roll i mappen given_files, som slumpar ett tal mellan 1 och 6.

Uppgift 4 - Herkules och hydran [2p]

Herkules försöker förgäves dräpa den fasansfulla hydran. Från början har hydran tre huvuden, men varje gång man hugger av ett så kommer det ut två nya! Skriv ett program som låter användaren mata in vad Herkules gör varje gång hydran sticker fram ett huvud. Användaren matar då in "HOPP" eller "HUGG". Vid "HOPP" hoppar Herkules bara undan från detta huvud, vid "HUGG" kapar han av huvudet, men det växer direkt ut två nya som sticks fram direkt!

KRAV: Använd rekursion. Ingar loopar tillåtna.

Körexempel 1:
```
Hydran sticker fram huvud 1: HOPP
Hydran sticker fram huvud 2: HOPP
Hydran sticker fram huvud 3: HOPP
```

Körexempel 2:
```
Hydran sticker fram huvud 1: HOPP
Hydran sticker fram huvud 2: HOPP
Hydran sticker fram huvud 3: HUGG
Hydran sticker fram huvud 3-1: HOPP
Hydran sticker fram huvud 3-2: HOPP
```

Körexempel 3:
```
Hydran sticker fram huvud 1: HUGG
Hydran sticker fram huvud 1-1: HOPP
Hydran sticker fram huvud 1-2: HOPP
Hydran sticker fram huvud 2: HOPP
Hydran sticker fram huvud 3: HUGG
Hydran sticker fram huvud 3-1: HUGG
Hydran sticker fram huvud 3-1-1: HOPP
Hydran sticker fram huvud 3-1-2: HUGG
Hydran sticker fram huvud 3-1-2-1: HOPP
Hydran sticker fram huvud 3-1-2-2: HOPP
Hydran sticker fram huvud 3-2: HOPP
```

Körexempel 4:
```
Hydran sticker fram huvud 1: HUGG
Hydran sticker fram huvud 1-1: HUGG
Hydran sticker fram huvud 1-1-1: HUGG
Hydran sticker fram huvud 1-1-1-1: HUGG
Hydran sticker fram huvud 1-1-1-1-1: HUGG
Hydran sticker fram huvud 1-1-1-1-1-1: HOPP
Hydran sticker fram huvud 1-1-1-1-1-2: HOPP
Hydran sticker fram huvud 1-1-1-1-2: HOPP
Hydran sticker fram huvud 1-1-1-2: HOPP
Hydran sticker fram huvud 1-1-2: HOPP
Hydran sticker fram huvud 1-2: HOPP
Hydran sticker fram huvud 2: HOPP
Hydran sticker fram huvud 3: HOPP
```