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	21:00	Betyg 3	Betyg G
2 poäng	20:30	Betyg 4	
2 poäng	19:30		Betyg VG
2 poäng	19:00	Betyg 5	
>=3 poäng	20:30	Betyg 5	
>=3 poäng	20:30		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 - Snödagbok [1p]

För varje dag som går i december så skriver gustav ner i sin dagbok om det snöade eller inte. Han vill nu ha ett program som tydligt visar hur snöigt det har varit. Gustav matar in '*' för en dag då det snöade och '-' för en dag då det inte snöade. Ditt program skall sedan mata ut andelen snöiga dagar (i procent) och skriva ut alla snöiga dagar för sig och alla uppehållsdagar för sig (se körexemplen):

KRAV: All inläsning skall göras i ett underprogram. Utskrift och eventuella beräkningar skall inte skötas i detta underprogram.

Körexempel 1:

```
Mata in dagar: *-*

**-
Det har varit 66.7% snöiga dagar.
```

Körexempel 2:

```
Mata in dagar: **-*-***-*---*

********------
Det har varit 57.1% snöiga dagar.
```

Körexempel 3:

```
Mata in dagar: *-----*-**-*-**-*-****---**----

**************----------------
Det har varit 45.2% snöiga dagar.
```

Uppgift 2 - Färger [1p]

På en målerifirma behövs ett datorsystem för att hålla rätt på många olika burkar målarfärg. I denna uppgift skall du skapa delar av detta system.

Du skall skapa en datatyp Color_Type som innehåller följande data:

- Namn (en textuell representation, t.ex. "BLUE", eller "DARKRED")
- Tre heltal som motsvara färgens RGB värde (d.v.s. hur pass stora delar av grundfärgerna rött, grönt och blått som färgen utgör). Heltalen ligger i intervallet [0, 255]

Till din datatyp skall du skapa ett underprogram Get som läser in till **en** variabel av typen Color_Type från tangentbordet. Du skall även skapa ett underprogram Put som skriver ut en Color_Type. Se körexempelet för in- resp utmatningsformatet. Exakt antal blanksteg i utskriften är inte viktigt.

Du skall skapa ett huvudprogram som läser in 6 st Color_Type och lagrar dessa i ett fält.

Du skall skapa ett underprogram Find_Brightest som söker igenom ett fält bestående av typen Color_Type och hittar den *ljusaste* färgen. Den ljusaste färgen är den färg vars summa av R, G och B som är högst. Låt huvudprogrammet anropa detta underprogram och skriva ut den ljusaste färgen.

Körexempel:

```
Welcome to the Color database!
Enter a color: BLUE    0 0 255
Enter a color: GREEN   20 255 20
Enter a color: DARKRED 128 0 0
Enter a color: GOLDEN  204 204 0
Enter a color: DEEPRPL 153 0 153
Enter a color: BLACK   0 0 0

The brightest color is GOLDEN  (R=204, G=204, B=0)
```

OBS: Användaren matar alltid in färgen med exakt 7 tecken (och fyller ut med blanksteg), som i exemplet ovan.

KRAV 1: Du skall använda en post för datatypen Color_Type;

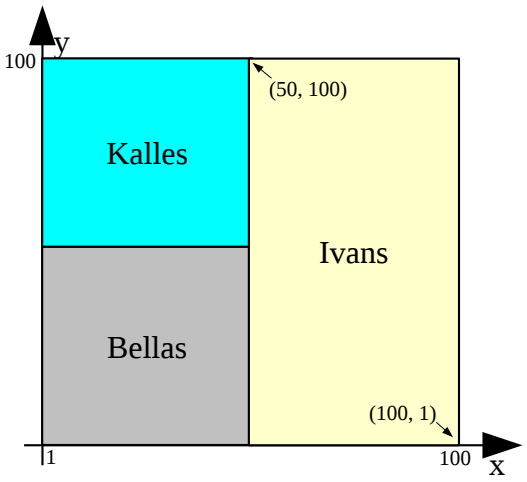
KRAV 2: Lös problemet generellt, det skall inte vara svårt att ändra koden så att det är fler än 6 färger som ska matas in.

Uppgift 3 - Sälja mark [2p]

Markägaren Maria har bestämt sig för att sälja av en bit av sin mark. Den bit som skall säljas är exakt 1000 x 1000 m stor, d.v.s. exakt 1 km² till ytan. Det har varit svårt att hita en enda köpare till marken och hon har därför tagit emot bud på mindre stycken. Hon har dock inte tillåtit några bud som inte är i hela tio meter. Minst möjliga köpbara yta är alltså 10 x 10 m.

Eftersom hon är noga med att se till att sålda bitar inte överlappar har hon skrivit ner koordinaterna för varje bit som hon har fått bud på. Om man ser hela marken som ett koordinatsystem där origo är nedre vänstra hörnet, Y-axeln går uppåt och X-axeln åt höger, så skriver hon ner koordinaterna för varje markstyckes övre vänstra hörn och nedre högra hörn (i enheten 10 m), så här:

```
1 100 49 50  Kalles
1 49 49 1    Bellas
50 100 100 1 Ivans
```



Även om Maria nu har sparat denna information på filen MARK.TXT så återstår ett problem: utan bilden så är det inte helt enkelt att se huruvida hela marken kommer bli såld. Din uppgift är att skapa ett program som läser igenom filen och sedan matar ut huruvida de markstycken som finns nedskrivna täcker en hel kvadratkilometer.

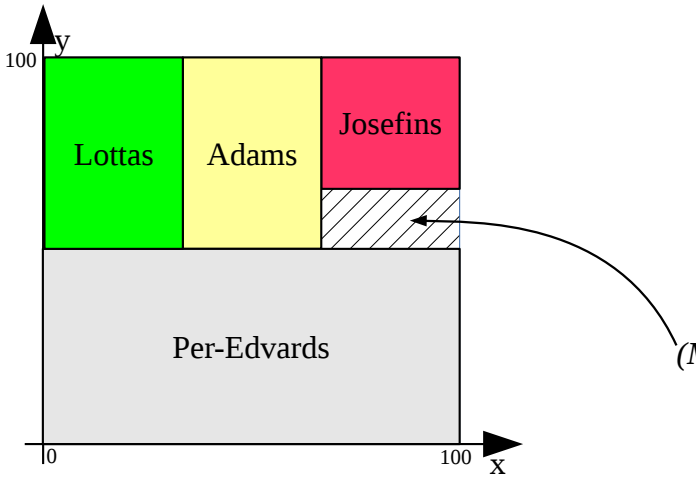
OBS: Innehållet i filen kan variera, men har alltid det formatet som beskrivs ovan. Det kan vara godtyckligt många rader i filen. Längden på strängen efter de fyra heltalen är maximalt 20 tecken. Du kan utgå ifrån att markstycken i filen aldrig överlappar.
TIPS: Använd ett fält. Tänk på att Maria bara tillåter markstycken med storlekar på hela 10 meter. Låtet fältet fungera som ett rutat papper, med 100x100 rutor.

Körexempel 1 (med MARK.TXT som ovan):

```
Markstyckena täcker exakt 1 km²!
```

Körexempel 2 (med MARK.TXT som här under):

```
Markstyckena täcker INTE hela marken.
```




```
67 100 100 67  Josefins
34 100 66 50   Adams
1 100 33 50    Lottas
1 49 100 1     Per-Edvards
```

(Mark som ej blir såld)

Uppgift 4 - Rymdstrid [2p]

I ett strategispel simuleras en strid mellan två rymdskepp med tärningsrullande. Två spelare rullar tärningar över flera rundor. Spelarna får ett antal tärningar var (beroende på hur många rymdskepp de vardera har). Rymdskeppen kan också vara olika utrustade, och är därmed olika bra. Varje spelare har därför en *nivå* - ett heltal som representerar skeppens förmåga (i intervallet [1, 6]). Spelaren måste slå över, eller lika med sin nivå för att skeppet skall träffa motståndaren i strid. Varje träff som spelarna få minskar moståndarens antal tärningar till nästa runda. Då en eller båda spelarna når 0 tärningar, är striden över.

Ett exempel: *Emma har 4 skepp med nivå 5, Karin har 2 skepp med nivå 3. Emma slår 1, 2, 2, 6 (d.v.s. 1 träff). Karin slår 3, 6 (d.v.s. två träffar). Emma har nu 2 skepp kvar, medan karin bara har 1. Emma slår 1, 4 (0 träffar). Karin slår en 5:a (1 träff). Emma blir av med ett skepp. I sista rundan slår Emma en 6:a (1 träff) och Karin slår en 2:a (ingen träff). Karin blir av med sitt skepp och har inga kvar, striden är över.*

Skriv ett program som låter användaren mata in hur många skepp de båda spelarna har och vilken nivå de har. Programmet skall sedan simulera förloppet ovan. Programmet skall skriva ut vilken spelare som vinner, eller om det blir oavgjort (då båda spelare har 0 skepp kvar).

Körexempel:

```
Mata in skepp spelare 1: 4
Mata in nivå spelare 1: 4
Mata in skepp spelare 2: 2
Mata in nivå spelare 2: 3

Runda 1:
Spelare 1: 2 6 2 1 (2 träffar)
Spelare 2: 3 6 (2 träffar)

Runda 2:
Spelare 1: 4 1 (0 träffar)
Spelare 2: 5 (1 träffar)

Runda 3:
Spelare 1: 6 (1 träffar)
Spelare 2: 2 (0 träffar)

Spelare 1 vinner!
```

KRAV: Inga loopar är tillåtna i din lösning. Använd rekursion.

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