2018-12-11 kl. 17-21

Ada - 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

Bonustid från Ada-laborationer tillkommer till dessa tidsgränser. Tiden för betyg 3/G överstiger dock inte fyra timmar.

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

2018-12-11 kl. 17-21

Uppgift 1 - Skonummer [1p]

Skriv ett program som ritar ut en skala med skonummer. Användaren matar in en undre och övre gräns. Programmet skall rita ut skonummer, enligt nedanstående körexempel. Du kan utgå ifrån att de inmatade talen ligger i intervallet [10, 99]. Inmatningen behöver ej felhanteras.

Körexempel 1:

```
Mata in undre gräns: 38
Mata in övre gräns : 44

--+--+--+--+--+--+--+
 38    40    42    44
    39    41    43
--+--+--+--+--+--+--+
```

Körexempel 2:

```
Mata in undre gräns: 39
Mata in övre gräns : 42

--+--+--+--+
    40    42
 39    41
--+--+--+--+
```

Körexempel 3:

```
Mata in undre gräns: 39
Mata in övre gräns : 43

--+--+--+--+--+
    40    42
 39    41    43
--+--+--+--+--+
```

Körexempel 4:

```
Mata in undre gräns: 34
Mata in övre gräns : 44

--+--+--+--+--+--+--+--+--+--+
 34    36    38    40    42    44
    35    37    39    41    43
--+--+--+--+--+--+--+--+--+--+
```

TIPS: Se upp för fullständig uppräkning, lös generellt...

2018-12-11 kl. 17-21

Uppgift 2 - Röstande Kamorfer [1p]

På planeten Kamorfia finns det sex politiska partier. Alla kamorfer tillhör ett politiskt parti men de är inte alltid helt lojala när det kommer till omröstningarna. Beroende på hur *populära* de olika partierna är så kan ett impopulärt parti faktiskt tappa röster till ett populärt parti.

På den givna filen *popular_vote.adb* finns lite given kod som låter användaren mata in partiernas popularitet och storlek (antal Kamorfer). Sedan simulerar programmet att alla partier med positiv popularitet *stjäl* röster från partier med negativ popularitet. Mängden stulna röster är två procentenheter per differens i popularitet.

För att programmet skall bli färdigt behöver du skapa datatypen *Demographic_Type* som skall vara ett sex element långt fält av poster. Varje post skall innehålla två heltal: *Popularity* och *Voters* (partiets storlek i antal röstande kamorfer). Till din datatyp behöver du även göra tre underprogram:
- Funktionen *Get_Popularity*, som tar två parametrar, en Demographic_Type D och ett heltal N, och returnerar det N:te partiets popularitet ur D.

- Funktionen *Get_Voters*, som tar två parametrar, en Demographic_Type D och ett heltal N, och returnerar det N:te partiets storlek ur D.

- Proceduren *Get* som läser in popularitet och storlek för varje parti från tangentbordet. Formatet är

```
P1 S1
P2 S2
P3 S3
P4 S4
P5 S5
P6 S6
```

Där PI är popularitet (ett heltal i intervallet [-5, 5]) för parti I och SI är storlek (ett heltal i intervallet [0, 100]) för parti I. Du kan utgå ifrån att användaren matar in datat korrekt. Ingen felhantering krävs.

Körexempel:

```
Mata in demografiskt data:
3 20
4 50
-3 60
0 50
0 10
-4 100

Parti 1 tar   7 röster från parti 3.
Parti 1 tar  14 röster från parti 6.
Parti 2 tar   8 röster från parti 3.
Parti 2 tar  16 röster från parti 6.
```

OBS: Du behöver inte göra ett paket för att hantera datatypen Demographic_Type.

KRAV: Du får inte ändra på det givna huvudprogrammet eller underprogrammet *Steal*.

2018-12-11 kl. 17-21

Uppgift 3 - Trettiosexiga klockslag [2p]

Skriv ett program som skriver ut alla *trettiosexiga* klockslag som finns under ett dygn. Vi definierar ett trettiosexigt klockslag som ett klockslag på formatet AB:CD:EF där A...F är siffror och summan

$$A + B + C + D + E + F = 36$$

Du har fått ett givet paket *Time_Handling* som innehåller datatyp och tillhörande underprogram för att hantera tider. Ditt huvudprogram får **endast** göra with/use på detta paket. Inga andra paket får användas.

Givetvis skall du lösa detta på ett generellt sätt, med loopar och/eller egna underprogram. Fullständig uppräkning är, som vanligt, inte godtagbart.

Körexempel:

```
08:59:59
09:49:59
09:58:59
09:59:49
09:59:58
17:59:59
18:49:59
18:58:59
18:59:49
18:59:58
19:39:59
19:48:59
19:49:49
19:49:58
19:57:59
19:58:49
19:58:58
19:59:39
19:59:48
19:59:57
```

2018-12-11 kl. 17-21

Uppgift 4 - Lufttryckstabell [2p]

Skriv ett program som skriver ut en tabell över lufttryck under en vecka. Programmet skall slumpa N heltal i intervallet [0, 1000] för varje dag i veckan måndag-söndag. Detta motsvarar N tider på dygnet. Användaren matar in N. N är ett godtyckligt positivt heltal.

Körexempel 1:

```
Mata in mätpunkter per dag: 1

LUFTTRYCKSTABELL
Dag 1    173 mbar
Dag 2    461 mbar
Dag 3    734 mbar
Dag 4    382 mbar
Dag 5    777 mbar
Dag 6    789 mbar
Dag 7    390 mbar
```

Körexempel 2:

```
Mata in mätpunkter per dag: 2

LUFTTRYCKSTABELL
Dag 1     35 mbar     99 mbar
Dag 2     21 mbar    112 mbar
Dag 3    335 mbar    608 mbar
Dag 4    990 mbar    845 mbar
Dag 5    756 mbar     51 mbar
Dag 6    872 mbar     78 mbar
Dag 7    166 mbar     54 mbar
```

Körexempel 3:

```
Mata in mätpunkter per dag: 5

LUFTTRYCKSTABELL
Dag 1    936 mbar    306 mbar    765 mbar    907 mbar    711 mbar
Dag 2    612 mbar    395 mbar    764 mbar    523 mbar    312 mbar
Dag 3    417 mbar    139 mbar    310 mbar    962 mbar    930 mbar
Dag 4    475 mbar    812 mbar    842 mbar    692 mbar    673 mbar
Dag 5    464 mbar    858 mbar    420 mbar    854 mbar    211 mbar
Dag 6    931 mbar    851 mbar    783 mbar     76 mbar    631 mbar
Dag 7    928 mbar     87 mbar    222 mbar    894 mbar     55 mbar
```

KRAV 1: Du skall använda **rekursion**. Inga loopar är tillåtna.

KRAV 2: Lös problemet generellt. Det skall inte vara svårt att ändra koden så att programmet funkar för t.ex. 8 dagar.