2018-12-11 kl. 13-17

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	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

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. 13-17

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 hela och halva skonummer, enligt nedanstående körexempel. Du kan utgå ifrån att de inmatade talen ligger i intervallet [10, 99.5] och att skalan har minst ett helnummer och ett halvnummer. Inmatningen behöver ej felhanteras.

Körexempel 1:

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

---+---+---+---+---+---+---+---+---+---+---+---+---+
  38 38.5 39 39.5 40 40.5 41 41.5 42 42.5 43 43.5 44
---+---+---+---+---+---+---+---+---+---+---+---+---+
```

Körexempel 2:

```
Mata in undre gräns: 39.5
Mata in övre gräns : 42

---+---+---+---+---+---+
 39.5 40 40.5 41 41.5 42
---+---+---+---+---+---+
```

Körexempel 3:

```
Mata in undre gräns: 37
Mata in övre gräns : 40.5

---+---+---+---+---+---+---+---+
  37 37.5 38 38.5 39 39.5 40 40.5
---+---+---+---+---+---+---+---+
```

Körexempel 4:

```
Mata in undre gräns: 37.5
Mata in övre gräns : 44.5

---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
 37.5 38 38.5 39 39.5 40 40.5 41 41.5 42 42.5 43 43.5 44 44.5
---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

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

2018-12-11 kl. 13-17

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*, en post som skall innehålla två delar: *Popularities* och *Voters* (partiernas storlekar i antal röstande kamorfer). Båda dessa är heltalsfält av längd sex. 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. 13-17

Uppgift 3 - Stigande klockslag [2p]

Skriv ett program som skriver ut alla *ökande* klockslag som finns under ett dygn. Vi definierar ett ökande klockslag som ett klockslag på formatet TT:MM:SS där T/M/S är siffror och där varje siffra i sekvensen är större än den föregående (till vänster).

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:

```
01:23:45
01:23:46
01:23:47
01:23:48
01:23:49
01:23:56
01:23:57
01:23:58
01:23:59
01:24:56
01:24:57
01:24:58
01:24:59
01:34:56
01:34:57
01:34:58
01:34:59
02:34:56
02:34:57
02:34:58
02:34:59
12:34:56
12:34:57
12:34:58
12:34:59
```

2018-12-11 kl. 13-17

Uppgift 4 - Luftfuktighetstabell [2p]

Skriv ett program som skriver ut en tabell över luftfuktighet under en vecka. Programmet skall slumpa N heltal i intervallet [0, 100] 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

LUFTFUKTIGHETSTABELL
-------------------------------------------------
  Dag 1  Dag 2  Dag 3  Dag 4  Dag 5  Dag 6  Dag 7
    98%     9%    33%    92%    62%    43%    86%
```

Körexempel 2:

```
Mata in mätpunkter per dag: 2

LUFTFUKTIGHETSTABELL
-------------------------------------------------
  Dag 1  Dag 2  Dag 3  Dag 4  Dag 5  Dag 6  Dag 7
     2%    93%     6%    21%    47%    54%    17%
    72%    20%    57%    55%    18%    35%    97%
```

Körexempel 3:

```
Mata in mätpunkter per dag: 7

LUFTFUKTIGHETSTABELL
-------------------------------------------------
  Dag 1  Dag 2  Dag 3  Dag 4  Dag 5  Dag 6  Dag 7
    66%    88%    99%    60%    57%    34%    63%
    57%    19%    31%    85%    93%    93%    27%
    79%     7%    64%    65%    71%    23%    83%
     9%    99%    51%    29%    26%    12%    88%
    90%    90%    38%    57%    92%     5%    67%
    24%    11%    50%    17%     9%    50%    48%
    60%    20%    96%    44%    47%    65%   100%
```

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.