O1.2: Enkel in- och utmatning

I denna inledande uppgift kommer du att öva på de grundläggande styrstrukturerna för repetition och val samt enkel in- och utmatning.

Mål

Du ska efter denna uppgift känna till och kunna använda

- hur man skriver ett litet Ada-program
- vad tecknet ”`;`” betyder och hur det används
- de tre formerna av repetitionssatsen `loop: loop`, `for` och `while`
- valsatsen `if`
- läsa in tal med `Get` (dock utan att ta hand om de fel som kan uppstå om användaren skriver in ett heltal då programmet läser ett reellt tal eller tecken)
- `Put` för att skriva ut både heltal, reella tal och strängar både med och utan formatdirektiv.

Uppgift

Skriv och testa momstabellprogrammet.

För denna uppgift gäller att snygga utskrifter ska åstadkommas med hjälp av formatdirektiv till `Put`, samt att du ska kunna motivera valet av iterationssats.

I programmet ska *felhantering* av användarens inmatning göras. Felhanteringen skall vara av typen rimlighetskontroll vilket innebär att användaren alltid kommer att mata in data av rätt datatyp. Detta ska göras så fort som felet ***kan*** upptäckas. Programmet ska ej avbrytas utan ny fråga ska ställas om användaren matar in ett felaktigt data! Tänk på att användaren kan mata in felaktiga data många gånger ...

Tänk på att ditt program måste ge en exakt korrekt utskrift jämfört med körexemplen. Inga extra rader, avsaknad av utskrifter, felaktiga värden, etc.

Momstabell

Konstruera ett program som skriver ut en momstabell. Programmet ska på terminalen fråga efter och ta som inmatning följande värden (där **alla** värden ska rimlighetskontrolleras! Tänk själv över vilka värden som är rimliga.):

- Nedre gräns för prisintervallet (decimaltal större än eller lika med 0.0)
- Övre gräns för prisintervallet (decimaltal)
- Steglängd i tabellen (decimaltal)
- Momsprocenten (uttryckt som decimaltal i intervallet [0%, 100%])

Användaren ska få upprepat mata in ett visst värde tills detta värde är OK enligt rimlighetskontrollen. Se körexemplen för detaljer kring hur detta ska fungera och se ut.

TIPS! För att få lite överblick kan det vara vettigt att börja med antagandet att användaren kommer att mata in rimliga värden och skriva ett program utan felkontroller för att sedan lägga till felkontrollerna först när programmet fungerar för rimliga indata. Programmet ska ge resultat enligt nedan vid körning (användarens indata ***kursiverad***):

Körexempel 1:

```
Första pris: 7.5
Sista pris: 12.0
Steg: 0.5
Momsprocent: 10.0

============ Momstabell ============
Pris utan moms  Moms   Pris med moms
     7.50       0.75        8.25
     8.00       0.80        8.80
     8.50       0.85        9.35
     9.00       0.90        9.90
     9.50       0.95       10.45
    10.00       1.00       11.00
    10.50       1.05       11.55
    11.00       1.10       12.10
    11.50       1.15       12.65
    12.00       1.20       13.20
```

Körexempel 2:

```
Första pris: 11.3
Sista pris: 12.0
Steg: 10.5
Momsprocent: 10.0

============ Momstabell ============
Pris utan moms  Moms   Pris med moms
    11.30       1.13       12.43
```

Körexempel 3:

```
Första pris: 10.00
Sista pris: 12.00
Steg: 0.3
Momsprocent: 20.00

============ Momstabell ============
Pris utan moms  Moms   Pris med moms
    10.00       2.00       12.00
    10.30       2.06       12.36
    10.60       2.12       12.72
    10.90       2.18       13.08
    11.20       2.24       13.44
    11.50       2.30       13.80
    11.80       2.36       14.16
```

OBS! Sista “Pris utan moms”-värdet i detta körexempel är 11.80 och inte 12.00 (inte heller 12.10)!

Körexempel 4:

```
Första pris: -1.23
Felaktigt värde!
Första pris: -3.00
Felaktigt värde!
Första pris: 4.0
Sista pris: 3.0
Felaktigt värde!
Sista pris: 6.2
Steg: -3.4
Felaktigt värde!
Steg: -1.0
Felaktigt värde!
Steg: 0.3
Momsprocent: 100.0

============ Momstabell ============
Pris utan moms  Moms   Pris med moms
     4.00       4.00        8.00
     4.30       4.30        8.60
     4.60       4.60        9.20
     4.90       4.90        9.80
     5.20       5.20       10.40
     5.50       5.50       11.00
     5.80       5.80       11.60
     6.10       6.10       12.20
```

Körexempel 5:

```
Första pris: 100.0
Sista pris: 101.0
Steg: 0.1
Momsprocent: 10.0

============ Momstabell ============
Pris utan moms  Moms   Pris med moms
   100.00      10.00      110.00
   100.10      10.01      110.11
   100.20      10.02      110.22
   100.30      10.03      110.33
   100.40      10.04      110.44
   100.50      10.05      110.55
   100.60      10.06      110.66
   100.70      10.07      110.77
   100.80      10.08      110.88
   100.90      10.09      110.99
   101.00      10.10      111.10
```

Körexempel 6 (När man arbetar med flyttal måste man tänka på att dessa inte är exakt lagrade i datorn. 0.1 i detta fall lagras med ett avrundat binärt värde. Hur skiljer detta körexempel sig ifrån körexempel 5 ovan?):

```
Första pris: 10.0
Sista pris: 11.0
Steg: 0.1
Momsprocent: 10.0

============ Momstabell ============
Pris utan moms  Moms   Pris med moms
    10.00       1.00       11.00
    10.10       1.01       11.11
    10.20       1.02       11.22
    10.30       1.03       11.33
    10.40       1.04       11.44
    10.50       1.05       11.55
    10.60       1.06       11.66
    10.70       1.07       11.77
    10.80       1.08       11.88
    10.90       1.09       11.99
    11.00       1.10       12.10
```

Några tips på testdata kan vara följande (kombinera lite olika för att se om ditt program verkar fungera). Tänk gärna ut fler själv.

Första pris:	**-1**	**0**	**10**	**100**				
Sista pris:	**-1**	**0**	**1**	**11**	**12**	**15**	**101**	**1000000**
Steg:	**-1**	**0**	**0.1**	**0.25**	**0.3**	**0.5**	**1**	**10**
Momsprocent:	**-1**	**0**	**1**	**20**	**50**	**100**	**101**	

Denna del är med för förståelse av avrundningsfel. Detta är inte ett krav för att du ska bli godkänd på uppgiften.

Tips för att räkna rader i ditt program. **(OBS! Detta är bara ett exempel på att detta är ett större problem än vad man kan tro. Detta körexempel kommer inte utföras i automaträttningen).** När du kör ditt program kan du låta utskriften av denna programkörning skrivas ut på en textfil istället för att det skrivs ut direkt i terminalen. För att göra detta startar du ditt program "som vanligt" men lägger till "> UTDATAFIL.TXT" efter den körbara filens namn. Exempel:

```
Terminalens prompt %./mitt_program > UTDATAFIL.TXT
1000.0
9000.0
0.01
10.0
```

Ditt program kommer köra som vanligt men all utskrift ifrån programmet kommer nu att skriva ut till textfilen UTDATAFIL.TXT istället för till skärmen (som vi är vana vid). Detta inkluderar även utskriften av "Första pris:", "Sista pris:" och så vidare, men programmet står fortfarande och väntar på din inmatning (som vanligt, alltså). Inmatningen ovan motsvarar alltså att första pris blir 1000.0, sista pris blir 9000.0, steg blir 0.01 och momsprocent blir 10.0.

På hemsidan finns en textfil med korrekt utskrift enligt ovan. Om du laddar ner den textfilen så kan du kontrollera om utskriften ifrån ditt program stämmer överens med denna genom att använda kommandot "diff". Utskriften markeras på samma sätt som i automaträttningen:

```
Terminalens prompt % diff UTDATAFIL.TXT KORREKT_UTDATA.TXT
```

Resultatet ifrån denna "diff" kommer antagligen att bli LÅNG (om utskriften ifrån ditt program inte är korrekt) så denna utskrift skriver vi inte ut här. Prata med assistentet om du vill ha mer information om detta.

För att räkna antalet rader i denna textfil kan du använda terminalkommandot "wc - l" (wc = WordCount, l som i "ett litet L") på textfilen:

```
Terminalens prompt % wc -l UTDATAFIL.TXT
800004 UTDATAFIL.TXT
```

Enligt ovanstående består alltså utskriften av 800,004 rader.

Körexempel 7 (Ett "extra intressant fall" (det finns många sådana) är följande. Detta testas ej i automaträttningen!):

```
Första pris: 0.05
Sista pris: 0.06
Steg: 0.01
Momsprocent: 10.0

============ Momstabell ============
Pris utan moms  Moms   Pris med moms
     0.05       0.01        0.05
```