Övningar till kapitel 1

1.1 Följande pseudokod beskriver en algoritm för hårtvätt:

```
Start hårtvätt
Blöt håret
SÅ LÄNGE håret känns smutsigt
    massera in shampo
    skölj
OM solen skiner
    låt håret självtorka
ANNARS
    använd hårtorken
Slut hårtvätt
```

a) Vilka delar av pseudokoden är *instruktioner*, vilka är *villkor* och vilka är *kontrollstrukturer*? Förklara ditt svar.
b) Dela in instruktionerna i huvud- och underinstruktioner.
c) Rita ett flödesschema till pseudokoden ovan.

1.2 Följande algoritm – *Kalle-algoritmen* – är formulerad på vanligt språk:

På vardagar går Kalle upp. Han tvättar sig, om mamman tittar på. På söndagar sover Kalle vidare tills mamman ropar honom till frukost, i så fall gör han som på vardagar.

Översätt flödesschemat till pseudokod.

1.3 Är följande pseudokod logiskt identisk med *Kalle-algoritmen* från övn 1.2 ?

```
Start Kanske_Kalle?
OM det är söndag
        sover Kalle vidare
    TILLS mamma ropar till frukost
ANNARS
    går han upp
OM mamma tittar på
    tvättar han sig
Slut Kanske_Kalle?
```

1.4 Rita ett flödesschema till *Kalle-algoritmen* från övn 1.2. Anta att lördag är en vardag.

Finns det i Kalle-algoritmen möjligheten till en evighetsloop?
När skulle den rent teoretiskt kunna inträffa?

1.5 Rita flödesschemat till följande pseudokod:

```
Sätt på radion
Välj en kanal och  lyssna
SÅ LÄNGE  du inte har hittat ett bra program
      byt kanal
      lyssna
Fortsätt att lyssna på det valda programmet
Stäng av radion
```

1.6 Rita ett flödesschema till följande pseudokod:

```
Start vinterklädsel_1
Läs av temperaturen
OM  temperatur <  0
      ta sjal, mössa och handskar
ANNARS
      OM  temperatur <  5
            ta sjal och mössa
      ANNARS
            OM  temperatur < 10
                  ta sjal
            ANNARS
                  slipper du vinterklädsel
Slut vinterklädsel_1
```

Använd dina kunskaper från *Programmering 1 och 2* för att koda pseudokoden ovan och flödesschemat till ett C# program. Läs in ett värde för temperatur och låt programmet avgöra val av klädsel genom att skriva ut "Ta ... ".

1.7 Samma algoritm som i övn 1.6 ovan kan formuleras med flervägsval:

```
Start vinterklädsel_2
Läs av temperaturen
VÄLJ  fall ur
      temperatur <  0: ta sjal, mössa och handskar
      temperatur <  5: ta sjal och mössa
      temperatur < 10: ta sjal
      Annars: slipper du vinterklädsel
Slut vinterklädsel_2
```

Rita flödesschemat till pseudokoden ovan och undersök den logiska likheten mellan flödesscheman i övn 1.6 och övn 1.7.

1.8 ***Collatz problemet*** * Rita flödesschemat till följande pseudokod:

```
Ta ett positivt heltal (startvärde)
Så länge talet ≠ 1 REPETERA:
   OM talet är udda
      multiplicera med 3, addera 1
   ANNARS
      dividera talet med 2
```

Att talföljderna i Collatz problemet slutar med 1 för *alla* startvärden, är matematiskt hittills obevisat. Men man kan testa: I appen **Mattekollen** kan du själv övertyga dig om att alla körningar slutar med 1 oavsett startvärde. Koden i appen är *Python* och inte *C#*. Ladda ned appen eller kör den som Webbapp: **app.mattekollen.se → En mobil pythonmiljö**. Du kan även komma åt webbappen direkt via adressen beta.mattekollen.se/#/app/coding

1.9 ***Collatz problemet (Inlämningsuppgift 1)*** Använd dina kunskaper från *Programmering 1 och 2* för att koda *Collatz problemet* i övn 1.8 till ett C# program. Lägg till pseudokoden: `Skriv ut talet` som sista instruktion i `Så länge`-satsen. Dvs skriv ut de talföljder som uppstår när man kör programmet.

Använd gärna flödesschemat du (förhoppningsvis) ritat i övn 1.7, annars titta på sid 212. För ≠ (inte lika med) kan du skriva C# koden **`!=`** och för att avgöra om **`tal`** är udda, koden **`if (tal % 2 == 1)`**. Testa programmet för startvärdena 3, 6, 7, 13 och 50. Testa även gärna större startvärden.

Studera de talföljder som uppstår. Fundera på varför alla startvärden slutar med 1 oavsett hur stora de är. Förmodan är att det är sant generellt, vilket dock hittills inte har kunnat bevisas matematiskt.

1.10 ***Kalkylatorn (Inlämningsuppgift 2)*** I detta projekt ska skapa en klass **`Calculator`** som stödjer följande funktionaliteter: addition, subtraktion, multiplikation, division och potentiering samt att kunna ange det största och minsta av två inmatade tal.

Dessutom ska din kalkylator vara igång kontinuerligt tills användaren väljer att stänga av den, vilket innebär att du måste lägga in en loop. De olika räkneoperationerna ska definieras i separata metoder och anropas i **`Main()`**.

Följande metoder ska definieras i klassen **`Calculator:`**

* Känt även som *Collatz förmodan* eller *(3n+1)-problemet* och kallat efter den tyske matematikern *Lothar Collatz* (1910-1990) som ställde upp det när han var student. Collatz var Professor för Tillämpad Matematik vid Hamburgs Universitet på 60-talet.

```
public double Add(double operand1, double operand2)
{
      // Additon av operand1 och operand2
}

public double Sub(double operand1, double operand2)
{
      // operand1 - operand2
      // Även subtraktion av negativa tal ska vara möjligt
}

public double Mult(double operand1, double operand2)
{
      // Multiplikation av parametrarna
}

public double Div(double operand1, double operand2)
{
      // operand1 / operand2
      // Division med 0 får ej förekomma (operand2 != 0)
}

public double Potens(double operand1, double operand2)
{
      // Beräkning av potens: operand1 upphöjt till operand2

}

public double max(double operand1, double operand2)
{
      // Returnera det större värdet av operand1 och operand2
      // Här kan du använda dig av den födefinierade metoden
      // Math.Max(double a, double b) för att snabbt
      // avgöra vilken av operanderna som är större
}

public double Min(double operand1, double operand2)
{
      // Returnera det mindre värdet av operand1 och operand2
      // Math.Min(double a, double b) kan användas
}
```

Programmet skall exekvera kontinuerligt tills användaren väljer att avsluta körningen. För att åstadkomma detta kan du exempelvis använda dig av en **do**-sats. Kalkylatorn kan avslutas genom att användaren matar in t.ex. tecknet ’q’ (Quit) istället för en operator.

Du får själv bestämma om du vill placera all kod i en fil eller om du hellre skapar en separat fil för klassen **Calculator** med alla ovannämnda metoder och en klass med **Main()** i en annan fil som testar klassen **Calculator**. Det senare är att föredra.

Det är upp till dig om du lägger in kod för att kunna hantera fel inmatning av operator eller andra felaktiga inmatningar.