# Laboration C++ 2 [G]

*Skickas in som Uppgift/Assignment 1.*

Skriv ett underprogram som sköter inmatning av ett tre flyttal A, B och C. Inmatningen skall vara felhanterad så att man garanterar att A + B >= C. Underprogrammet skall endast ha tre parametrar (de flyttal som man läser in till). Anropa ditt underprogram från huvudprogrammet. Huvudprogrammet skall sköta utskriften av olikheten.

**OBS:** Tänk noga över huruvida parametrar är in- eller utdata. Parametrar som endast är indata till en funktion skall deklareras med *const*.

**Körexempel 1**

```
Mata in A, B och C: 3.5 5.5 9.0
Du matade in korrekt!
3.5 + 5.5 >= 9.0
```

**Körexempel 2**

```
Mata in A, B och C: 5.6 2.4 10.0
Fel! A + B >= C: 1.0 1.0 3.0
Fel! A + B >= C: 0.0 0.0 1.0
Fel! A + B >= C: 10.5 10.5 19.6
Du matade in korrekt!
10.5 + 10.5 >= 19.6
```

# Laboration C++ 2 [VG]

*Skickas in som Uppgift/Assignment 2.*

Givet huvudprogrammet nedan, skriv funktionerna *set_first(S, C)* och *set_index(S, I, C)* som fungerar på följande vis.

- set_first tar två parametrar, strängen S och tecknet C. Första tecknet i strängen S skall sättas till C. Funktionen returnerar orginalvärdet på S.
- set_index tar tre parametrar, strängen S, heltalet I och tecknet C. Tecknet på plats I i strängen skall sättas till C. Du kan anta att I är icke-negativ och kortare än längden på S. Funktionen returnerar orginalvärdet på S.
- set_first skall anropa set_index för att lösa sin uppgift så att duplicering av kod undviks.

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4  int main()
5  {
6    string s{"Malin"};
7    cout << set_first(s, 'B') << " - "
8         << set_index(s, 3, 'o') << " - "
9         << set_index(s, 4, 'o') << " - "
10        << s << endl;
11   return 0;
12 }
```

**OBS:** Tänk noga över huruvida parametrar är in- eller utdata. Parametrar som endast är indata till en funktion skall deklareras med *const*.

**TIPS:** Strängar indexeras från 0 i C++. Funktionen *.at(n)* kan användas för att komma åt plats *n* i en sträng. T.ex:

```
1    string s{"ebba"};
2    cout << s.at(1); // skriver ut 'b';
3    s.at(3) = 'e';   // nu är s "ebbe"
```

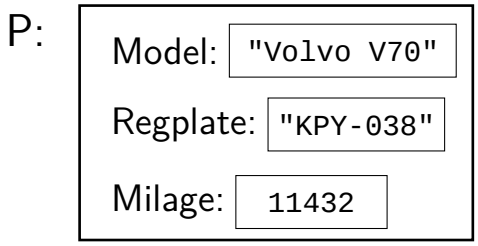
### Körexempel 1

```
Malin - Balin - Balon - Baloo
```

# Laboration C++ 3 [G]

*Skickas in som Uppgift/Assignment 3.*

Definiera en ny typ enligt figuren nedan. Din typ skall vara en *struct* och innehålla de delar som figuren visar. Datat i varje delvariabel visar vilken typ som den delvariabeln skall vara.




Skapa sedan ett huvudprogram som deklararerar en variabel P av din nya typ. Programmet skall sedan göra följande:

1. Låta användaren mata in data som lagras i P.
2. Deklarera en variabel av typen *vector*. Vektorns inre typ skall vara den struct-typ som du definierat.
3. Stoppa in P på första plats i vektorn.
4. Slutligen skall programmet skriva ut datat som ligger lagrat i vektorns första element **(du skall inte använda variabeln P i detta steg)**.

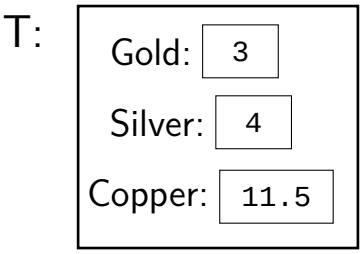
**Körexempel 1**

```
Mata in modell: Volvo V70
Mata in registreringsummer: KPY-038
Mata in mätarställning: 11432

Du matade in Volvo V70, KPY-038, 11432
```

# Laboration C++ 3 [VG]

*Skickas in som Uppgift/Assignment 4.*




Skapa ett program där du definierar ett fält (använd typen *vector*) av typen T som visas ovan. T representerar en hjältes pengainnehav och skall vara en *struct*. Låt sedan användaren fylla på med data i denna datastruktur (enligt körexemplet). Inmatningen avslutas med enter och ctrl-d. Slutligen skall programmet gå igenom och räkna hur många hjältar som har mer kopparmynt än silvermynt **och** mer silvermynt än guldmynt (i exemplet nedan är det bara de två första raderna som uppfyller detta villkor).

**Körexempel 1**

```
Mata in hjätars pengar (avsluta med enter och Ctrl-D):
3 4 11.5
1 2 3.5
2 1 2.15
1 1 2.99
3 3 5.5
[ctrl-d]

2 hjältar har mer kopparmynt än silver och mer silver än guld.
```

# Laboration C++ 4 [G]

*Skickas in som Uppgift/Assignment 5.*

På filen WORDS.TXT ligger det många ord över flera rader. Skapa ett program som läser igenom filen och skriver ut varje rads längd. Det är tillåtet att läsa in hela rader till programmet med t.ex. getline. Filen kan ha godtyckligt många och godtyckligt långa rader, alltså är det inte tillåtet att läsa in hela filen till minnet samtidigt.

**Körexempel 1**

```
12 tecken
11 tecken
10 tecken
9 tecken
3 tecken
```

# Laboration C++ 4 [VG]

*Skickas in som Uppgift/Assignment 6.*

På filen DATA.TXT ligger det rader med följande format:

```
I C W F
```

Där I är ett heltal, C är ett tecken, W är ett ord och F är ett flyttal. Allt data är separerat med ett blanksteg. Tyvärr har det smugit sig in rader i filen som inte följer detta format! Följande är ett exempel på hur filen kan se ut:

```
1 752 u grazi 4.55
2 0 t meltborn 0.5
3 y vagga 45
4 19 9 turok 3.14
5 1 1 1 1
6 -24 ! beleriand xy
7 kortrad
8 -5 kortis
```

Skapa ett program som läser igenom filen och skriver ut de rader som inte följer formatet. Det är tillåtet att läsa in hela rader till programmet med t.ex. getline. Filen kan ha godtyckligt många och godtyckligt långa rader, alltså är det inte tillåtet att läsa in hela filen till minnet samtidigt.

**TIPS:** Läs in raden till en sträng, använd sedan en *stringstream* som du försöker läsa med formaterad inmatning ifrån.

**Körexempel 1**

```
Fel format: y vagga 45
Fel format: -24 ! beleriand xy
Fel format: kortrad
Fel format: -5 kortis
```