Laboration 4 [G]

Skickas in som Uppgift/Assignment 1.

Givet huvudprogrammet nedan (som även finns på filen test_make_price.adb), skapa det paket som behövs (Price_Handling) d.v.s. en .ads- och en .adb-fil. Paketet skall innehålla datatypen Price_Type, och underprogrammen *Make_Price* och *Put* för denna. *Make_Price* skall ta två heltal och returnera motsvarande pris (kronor och ören) som en Price_Type. Put skall även ta en parameter *Separator*, ett tecken som anger hur man vill separera kronorna från örena (se körexemplet). Din datatyp skall vara en post och skall vara privat.

```
1   with Ada.Text_IO;                use Ada.Text_IO;
2   with Price_Handling;             use Price_Handling;
3   procedure Test_Make_Price is
4
5     P1, P2 : Price_Type;  -- En typ för att lagra ett pris
6                           -- i hela kronor och ören
7   begin
8     P1 := Make_Price(19, 50);
9     P2 := Make_Price(10, 00);
10    Put_Line("Första priset: ");
11    Put(P1, Separator => ':'); -- Skriver ut priset med ':' emellan
12    New_Line;
13    Put_Line("Andra priset: ");
14    Put(P2, Separator => ','); -- Skriver ut priset med ',' emellan
15    New_Line;
16  end Test_Make_Price;
```

Körexempel 1

```
Första priset:
19:50
Andra priset:
10,00
```

TIPS: Du behöver inte sätta något default/skönsvärde på parametern separator i Put. Vi antar att den som anropar alltid har med två parametrar.

Laboration 4 [VG]

Givet huvudprogrammet nedan (som även finns på filen test_reg_plates_next.adb), skapa det paket som behövs (Reg_Plate_Handling) d.v.s. en .ads- och en .adb-fil. Paketet skall innehålla datatypen Reg_Plate_Type, och underprogrammen *Put* och *Next_Plate* för denna. *Next_Plate* skall ta fram nästa nummer i ordningen. När man har ”tickat” upp till nummer 999 på en plåt så är nästa nummer 000 och man ökar på den sista bokstaven istället. Ett exempel är AAA-999, där nästa skulle bli AAB-000. Nästa efter AAZ-999 skulle bli ABA-000. Observera att vi bara använder stora bokstäver ’A’-’Z’. Du behöver inte lösa vilken som är nästa efter ZZZ-999. Din datatyp skall vara en post men behöver inte vara privat.

```
1  with Ada.Text_IO;                use Ada.Text_IO;
2  with Reg_Plate_Handling;         use Reg_Plate_Handling;
3  procedure Test_Reg_Plate_Next is
4
5    R : Reg_Plate_Type;  -- En typ för att lagra ett registrerings-
6                         -- nummer för en bil. Har Tre tecken och
7                         -- ett (tresiffrigt) heltal.
8  begin
9    R.Letters := "KPY";
10   R.Numbers := 038;
11   Put("R är ");
12   Put(R);
13   New_Line;
14   Put("Nästa registreringsnummer är ");
15   Put(Next_Plate(R));
16 end Test_Reg_Plate_Next;
```

Körexempel 1

```
R är KPY-038
Nästa registreringsnummer är KPY-039
```

TIPS: Tecken har en inbördes ordning i ASCII-tabellen. Man kan få fram ett teckens nummer med funktionen Character’Pos. Man kan få fram motsvarande tecken från ett nummer med Character’Val.

```
1    I : Integer;
2    C : Character := 'A';
3  begin
4    I := Character'Pos(C);   -- I blir 65
5    C := Character'Val(66);  -- C blir 'B'
```

TIPS 2: Testa noga. T.ex. med KPY-999, KPZ-999 och KZZ-999.

Laboration 5 [G]

Skickas in som Uppgift/Assignment 2.

Skriv ett program som låter användaren mata in ett heltal N och en sträng med tre bokstäver. Programmet skall sedan skriva ut strängen N gånger enligt körexemplen.

KRAV: Lös med rekursion. Inga loopar är tillåtna.

Körexempel 1

```
Mata in ett heltal och en sträng (3 tecken): 3 KUL
3. KUL
2. KUL
1. KUL
```

Körexempel 2

```
Mata in ett heltal och en sträng (3 tecken): 7 Ja!
7. Ja!
6. Ja!
5. Ja!
4. Ja!
3. Ja!
2. Ja!
1. Ja!
```

Körexempel 3

```
Mata in ett heltal och en sträng (3 tecken): 0 Tur
```

Laboration 5 [VG]

Skickas in som Uppgift/Assignment 3.

I paketet Linked_List (".ads" och ".adb" finns givna) ligger datatypen List_Type definierad. Det är en länkad lista av heltal. Paketet har även en procedur Build_Test_List som kan bygga upp en testlista och en procedur Put som kan skriva ut hela listan. Lägg till underprogrammet *Equal* till paketet. Underprogrammet skall ta två listor som parametrar A och B och jämföra dem elementvis. Underprogrammet skall returnera sant om elementen i A stämmer helt överens med elementen i B, och falskt annars. Du kan utgå ifrån att listorna är lika långa. Man ska alltså kunna köra följande huvudprogram:

```
1  with Ada.Text_IO;                use Ada.Text_IO;
2  with Ada.Integer_Text_IO;        use Ada.Integer_Text_IO;
3  with Linked_List;                use Linked_List;
4
5  procedure Main_Equal is
6    L1, L2, L3 : List_Type;
7  begin
8    Build_Test_List(L1); -- bygger listan 5 -> 10 -> 15
9    Build_Test_List(L2); -- bygger listan 5 -> 10 -> 15
10   Build_Test_List(L3, 3); -- bygger listan 3 -> 6 -> 9
11   if Equal(L1, L2) then
12     Put_Line("L1 och L2 var lika.");
13   else
14     Put_Line("L1 och L2 var INTE lika???");
15   end if;
16   if Equal(L2, L3) then
17     Put_Line("L2 och L3 VAR lika???");
18   else
19     Put_Line("L2 och L3 var inte lika.");
20   end if;
21 end Main_Equal;
```

KRAV: Lös med rekursion. Inga loopar är tillåtna.

Körexempel 1

```
L1 och L2 var lika.
L2 och L3 var inte lika.
```