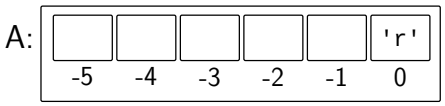
Laboration 3 [G]

Skickas in som Uppgift/Assignment 1.

Skriv ett program som definierar en fältdatatyp som motsvarar bilden nedan (värdet i fältet visar vilken inre datatyp som fältet har). Deklarera sedan en variabel A av denna typ. A skall sedan fyllas med data som användaren får mata in. Programmet skall slutligen skriva ut fältets innehåll (enligt nedanstående körexempel).



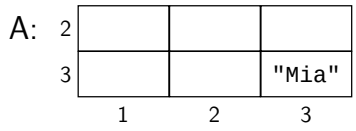
Körexempel 1

```
Mata in data: castor
Följande data finns i fältet: 'c' 'a' 's' 't' 'o' 'r'
```

Laboration 3 [VG]

Skickas in som Uppgift/Assignment 2.

Skriv ett program som definierar en fältdatatyp som motsvarar bilden nedan (värdet i fältet visar vilken inre datatyp som fältet har). Deklarera sedan en variabel A av denna typ. A skall sedan fyllas med data som användaren får mata in. Slutligen skall programmet gå igenom fältet och skriva ut det baklänges (enl. körexemplet).



Körexempel 1

```
Mata in data: Liv Eva Mia
Gus Joe Pam

Mia Eva Liv
Pam Joe Gus
```

Laboration 4 [G]

Skickas in som Uppgift/Assignment 3.

Givet huvudprogrammet nedan (som även finns på filen test_price_plus.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 *Put* och operatorn + för denna. Din datatyp behöver inte vara privat.

```
1  with Ada.Text_IO;                use Ada.Text_IO;
2  with Price_Handling;             use Price_Handling;
3  procedure Test_Price_Plus is
4
5    P1, P2 : Price_Type;  -- En typ för att lagra ett pris
6                          -- i hela kronor och ören
7  begin
8    P1.Kronor := 19;
9    P1.Oren   := 50;
10   P2 := (Kronor => 10, Oren => 0);
11   Put("P1 är ");
12   Put(P1);
13   New_Line;
14   Put("P2 är ");
15   Put(P2);  -- Om örena är 0 så skall ";-" skrivas ut istället
16   New_Line;
17
18   Put("Om vi adderar P1 och P2 får vi ");
19   Put(P1 + P2);
20   New_Line;
21 end Test_Price_Plus;
```

Körexempel 1

```
P1 är 19;50
P2 är 10;-
Om vi adderar P1 och P2 får vi 29;50
```

Laboration 4 [VG]

Skickas in som Uppgift/Assignment 4.

Givet huvudprogrammet nedan (som även finns på filen test_price_minus.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 *Put* och operatorn - för denna. Om det resulterande priset i operatorn skulle vara negativt så skall undantaget Price_Underflow kastas (resas). Ditt paket skall inte fånga några undantag. Din datatyp behöver inte vara privat.

```
1  with Ada.Text_IO;                use Ada.Text_IO;
2  with Price_Handling;             use Price_Handling;
3  procedure Test_Price_Minus is
4
5    P1, P2 : Price_Type;  -- En typ för att lagra ett pris
6                          -- i hela kronor och ören
7  begin
8    P1.Kronor := 19;
9    P1.Oren   := 50;
10   P2 := (Kronor => 10, Oren => 0);
11   Put("P1 är ");
12   Put(P1);
13   New_Line;
14   Put("P2 är ");
15   Put(P2);  -- Om örena är 0 så skall ";-" skrivas ut istället
16   New_Line;
17
18   Put("Om vi subtraherar P2 från P1 får vi ");
19   Put(P1 - P2);
20   New_Line;
21   Put("Om vi subtraherar P1 från P2 får vi ");
22   Put(P2 - P1);
23   New_Line;
24 exception
25   when Price_Underflow =>
26     New_Line;
27     Put_Line("Resultatet av en subtraktion blev negativ!");
28 end Test_Price_Minus;
```

Körexempel 1

```
P1 är 19;50
P2 är 10;-
Om vi subtraherar P2 från P1 får vi 9;50
Om vi subtraherar P1 från P2 får vi
Resultatet av en subtraktion blev negativ!
```

Laboration 5 [G]

Skickas in som Uppgift/Assignment 5.

Skriv ett program som låter användaren mata in N och beräknar produkten av alla tal i intervallet [-10, N]. N ligger i intervallet [-10, -1].

KRAV: Lös med rekursion. Inga loopar är tillåtna.

Körexempel 1

```
Mata in ett tal: -10
Produkten blev -10
```

Körexempel 2

```
Mata in ett tal: -9
Produkten blev 90
```

Körexempel 3

```
Mata in ett tal: -2
Produkten blev -3628800
```

Laboration 5 [VG]

Skickas in som Uppgift/Assignment 6.

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 *Find* till paketet. Underprogrammet skall ta en lista och ett heltal D som parametrar och söka efter den första förekomsten av talet D i listan. Find skall sedan returnera en Lista som börjar med det sökta elementet. Observera att Find inte skall kopiera något data. Du kan utgå ifrån att det sökta elementet alltid finns i listan. Följande huvudprogram skall sedan gå att köra:

```
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_Find is
6    D    : Integer;
7    L, P : List_Type;
8  begin
9    Build_Test_List(L); -- bygger listan 5 -> 10 -> 15
10   Put(L);
11   Put("Vilket element vill du söka efter? ");
12   Get(D);
13   P := Find(L, D);
14   Put(P);
15 end Main_Find;
```

KRAV: Lös med rekursion. Inga loopar är tillåtna.

Körexempel 1

```
Listan: 5 -> 10 -> 15
Vilket element vill du söka efter? 15
Listan: 15
```

Körexempel 2

```
Listan: 5 -> 10 -> 15
Vilket element vill du söka efter? 10
Listan: 10 -> 15
```