

***:96 Overheads**

**Part 2bx: Encoding using
ASN.1 - solutions to exercises**

Last update: 2000-10-09 18:24

Exercise 1 Solution

Solution alternative 1 to Exercise 1

```
ScaleReading ::= [APPLICATION 0] SEQUENCE { weight Weight,  
                                              itemno Itemno  
                                              }  
Weight ::= [APPLICATION 1] REAL - - in grams  
Itemno ::= [APPLICATION 2] INTEGER
```

Solution alternative 2 to Exercise 1

```
ScaleReading ::= [APPLICATION 0] SEQUENCE {  
    weight REAL, - - in grams  
    itemno INTEGER  
}
```

continued on the next page

Warning: The use of the **APPLICATION** tag is not recommended in the 1994 version of ASN.1. So with the 1994 style of ASN.1:

Solution alternative 3 to Exercise 1

```
ScaleReading ::= SEQUENCE { weight Weight,  
                                itemno Itemno  
                                }
```

```
Weight ::= REAL - - in grams
```

```
Itemno ::= INTEGER
```

Exercise 2 Solution

```
Box ::= SEQUENCE{  
    height Measurement,  
    width Measurement,  
    length Measurement  
}
```

```
Measurement ::= SEQUENCE {  
    yards INTEGER,  
    feet INTEGER,  
    inches REAL
```

Exercise 3 Solution

```
Measurement ::= SEQUENCE {  
    yards INTEGER,  
    feet INTEGER (0 .. 2),  
    inches INTEGER (0 .. 1199)  
}
```

Exercise 4 Solution

```
Voter ::= SEQUENCE {  
  vote Vote,  
  age Age,  
  gender Gender  
}
```

```
Vote ::= INTEGER {  
  labour(0),  
  liberals (1),  
  conservatives (2),  
  other (3)  
} (0 .. 3)
```

Alternative definition of "Vote":

```
Vote ::= ENUMERATED {  
  labour(0),  
  liberals (1),  
  conservatives (2),  
  other (3)  
}
```

ASN.1 experts say that the following is not permitted

```
Male ::= [APPLICATION  
  4] BOOLEAN {  
  male (TRUE),  
  female (FALSE)  
}
```

Age ::= INTEGER (18 .. MAX)

Gender ::= BOOLEAN or

Gender ::= ENUMERATED [male(0), female(1) }

Exercise 5 Solution

```
HomeTownVoter ::= SEQUENCE {  
    hometownvote Sthvote,  
    age Age,  
    sex Sex  
}
```

```
Hometomwnvote ::= INTEGER  
    liberals (1),  
    conservatives (2),  
    other (3),  
    hometown-party (4),  
    drivers-party (5)  
( 0 .. 5 )
```

Note, some people claim that it would be allowed to write:

```
}( INCLUDES Vote | 4 | 5 )
```

as the last line above, but other people claim this is not allowed.

Exercise 6 Solution

Solution alternative 1 to Exercise 6:

Secrecy ::= INTEGER { open(1), secret(2), topsecret(3) } (1..3)

Solution alternative 2 to Exercise 6:

(better according to ASN.1 experts)

Secrecy ::= ENUMERATED { open(1), secret(2), topsecret(3) }

Exercise 7 Solution

Solution alternative 1 to Exercise 7:

```
StabSecrecy ::= INTEGER { open(1), secret(2), topsecret(3),  
    extratopsecret(4) }  
    (INCLUDES Secrecy | 4 )
```

Solution alternative 2 to Exercise 7:

(better according to ASN.1 experts)

```
StabSecrecy ::= ENUMERATED { open(1), secret(2),  
    topsecret(3), extratopsecret(4) }
```

Exercise 8 Solution

Solution alternative 1 to Exercise 8

```
Pattern ::= SEQUENCE {  
    height INTEGER,  
    width INTEGER,  
    pattern BIT STRING - - row by row  
}
```

Solution alternative 2 to Exercise 8

```
Row ::= BIT STRING
```

```
Pattern ::= SEQUENCE {  
    height INTEGER,  
    width INTEGER,  
    pattern SEQUENCE OF Row  
}
```

Exercise 9 Solution

```
InStore ::= BIT STRING {  
    a3 (0),  
    a4 (1),  
    a5 (2),  
    a6 (3)  
} (SIZE(4))
```

Exercise 10 Solution

Solution taken from X.411, 1998 version

ub-organization-name-length INTEGER ::= 64

**OrganizationName ::= PrintableString
(SIZE (1 .. ub-organization-name-length))**

Solution, using new constructs from the 1994 version of ASN.1:

**Name {INTEGER : name-length} ::= PrintableString
(Size(1..name-length))**

OrganizationDirectorName ::= Name {64}

Exercise 11 Solution

Solution alternative 1 to Exercise 11

```
PersonRecord ::= SET {  
    pnumber Pnumber,  
    name Nametype OPTIONAL,  
    income Incometype OPTIONAL  
}  
  
Pnumber1 ::= [APPLICATION 1] PrintableString  
    (FROM ("0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" | "-" | "  
    "))  
  
Pnumber ::= Pnumber1 (SIZE (13))  
Nametype ::= GeneralString (SIZE (1 .. 40))  
Incometype ::= INTEGER (0 .. MAX)
```

Solution alternative 2 to Exercise 11

```
PersonRecord ::= SET {
```

```

    pnumber Pnumber,
    name Nametype OPTIONAL,
    income Incometype OPTIONAL
}

```

continued

```

Pnumber1 ::= PrintableString (FROM ("0" | "1" | "2" | "3" | "4"|"5" | "6" | "7" | "8" |
    "9" | "-" | " "))

```

```

Pnumber ::= Pnumber1 (SIZE (13))

```

```

Nametype ::= GeneralString (SIZE (1 .. 40))

```

```

Incometype ::= INTEGER (0 .. MAX)

```

Solution alternative 3 to Exercise 11

```

Pnumber1 ::= PrintableString (FROM ("0" | "1" | "2" | "3" | "4"|"5" | "6" | "7" | "8" |
    "9" | "-" | " "))

```

```

PersonRecord ::= [APPLICATION 0] SET {
    pnumber Pnumber1 (SIZE (13))
    name GeneralString (SIZE (1 .. 40)) OPTIONAL,
    income INTEGER (0 .. MAX) OPTIONAL
}

```

Note: With the 1994 version of ASN.1, you might also write:

```

Pnumber1 ::= PrintableString (FROM ("0" .. "9" | "-" | " "))

```

Exercise 12 Solution

Solution alternative 1 to Exercise 12

```
PersonRecord ::= SET {  
    pnumber Pnumber,  
    gname GNametype OPTIONAL,  
    sname SNametype OPTIONAL,  
    Income Incometype OPTIONAL  
}  
  
Pnumber1 ::= PrintableString  
    (FROM ("0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" |  
    "-" | " "))  
  
Pnumber ::= Pnumber1 (SIZE (13))  
GNametype ::= [APPLICATION 0] GeneralString (SIZE (1 .. 40))  
SNametype ::= GeneralString (SIZE (1 .. 40))  
Incometype ::= INTEGER (0 .. MAX)
```

Solution alternative 2 to Exercise 12

```
PersonRecord ::= SET {  
    pnumber Pnumber,  
    name Nametype OPTIONAL,  
    income Incometype OPTIONAL  
}  
  
Pnumber1 ::= PrintableString  
    (FROM ("0" | "1" | "2" | "3" | "4"|"5" | "6" | "7"  
    | "8" | "9" | "-" | " "))  
  
Pnumber ::= Pnumber1 (SIZE (13))  
  
Nametype ::= SEQUENCE {  
    sName GeneralString (SIZE (1 .. 40)),  
    gName GeneralString (SIZE(1 .. 40))  
}  
  
Incometype ::= [APPLICATION 3] INTEGER (0 .. MAX)
```


Question: Why is the solution below not correct?

```
PersonRecord ::= [APPLICATION 0] SET {  
    pnumber Pnumber,  
    gname Nametype OPTIONAL,  
    sname Nametype OPTIONAL,  
    income Incometype OPTIONAL  
    }  
Pnumber1 ::= PrintableString  
    (FROM ("0" | "1" | "2" | "3" | "4" | "5" |  
    "6" | "7" | "8" | "9" | "-" | " "))  
Pnumber ::= Pnumber1 (SIZE (13))  
Nametype ::= GeneralString (SIZE (1 .. 40))  
Incometype ::= INTEGER (0 .. MAX)
```

Answer: The receiving computers cannot know if a name with only one component is only a **gname** or only a **sname**.

Exercise 13 Solution

```
PositiveCoordinate ::= XYCoordinate  
    ( WITH COMPONENTS {  
      x (0 .. MAX)  
      y (0 .. MAX)  
    })
```

Exercise 14 Solution

Solution alternative 1 to Exercise 14

```
AnonymousMessage ::= Message  
    ( WITH COMPONENTS { ... , author  
    ABSENT }  
    )
```

Solution alternative 2 to Exercise 14

```
AnonymousMessage ::= Message  
    ( WITH COMPONENTS {  
    author ABSENT,  
    textbody }
```

Exercise 15 Solution

```
FullName ::= SEQUENCE {  
    givenName [0] IA5String OPTIONAL,  
    initials [1] IA5String OPTIONAL,  
    surname [2] IA5String  
}
```

Question: Can the tags in the solution above be removed?

Yes, you can always remove one of the tags, since it will then get the **UNIVERSAL** tag of **IA5String**, which is different than the other user-defined tags.

If you have **AUTOMATIC** tagging set, you can remove all the tags. Otherwise, two of them must be kept, since the elements must have different tags to separate them. If the first two elements had not been **OPTIONAL**, then the tags would not have been required, since then the elements could be separated by their order in the **SEQUENCE**.

If you reorder the element with surname first, you can skip the tags on two of the elements, since then surname can be recognized by the order of the elements.

Exercise 16 Solution

Solution alternative 1 to Exercise 16

```
BasicFamily ::= SEQUENCE {  
    husband [0] IA5String OPTIONAL,  
    wife [1] IA5String OPTIONAL,  
    children [2] SEQUENCE OF IA5String  
    OPTIONAL  
}
```

With automatic tagging, the tags above can be removed.

Question: Is **SEQUENCE OF** or **SET OF** best in this exercise?

Answer: If you want to indicate the order of birth the children, **SEQUENCE OF** is better.

Exercise 17 Solution

```
ChildLessFamily ::= BasicFamily  
    ( WITH COMPONENTS {  
      ... , children ABSENT  
    }  
  )
```

Exercise 18 Solution

```
Vessel ::= CHOICE {  
    aircraft Aircraft,  
    ship Ship,  
    train Train,  
    motorcar MotorCar }
```

Exercise 19 Solution

Solution alternative 1 to Exercise 19

```
GeneralNameListA ::= gs < NameListA
GeneralNameListB ::= NamelistB
                    ( WITH COMPONENT
                      (WITH COMPONENTS {gs} )
                    )
```

Solution alternative 2 to Exercise 19

```
GeneralNameListA ::= NameListA ( WITH COMPONENTS {gs} )
GeneralNameListB ::= NamelistB
                    ( WITH COMPONENT
                      (WITH COMPONENTS {gs} ) )
```


Exercise 20a Solution

```
Vote ::= SEQUENCE {  
    voterName IA5String,  
    votevalue CHOICE {  
        chosenAlternative AlternativeNumber,  
        setvalue SET OF SEQUENCE {  
            alternative AlternativeNumber,  
            score INTEGER ( 0 .. 10 )  
        }  
    }  
}
```

Exercise 20b Solution

```
vote          = voter-name ","
               (One-choice / Choice-list )
voter-name    = "\"" name "\""
name          = 1*namechar
namechar      = <any printable ASCII character
               except "\"">
One-choice    = "Single:" 1*DIGIT
Choice-list   = "Multiple:" 1#(alternative LWSP
               score)
alternative   = 1*DIGIT
Score         = "0" / "1" / "2" / "3" / "4" / "5" /
               "6" / "7" / "8" / "9" / "10"
```

Exercise 21 Solution

```
WeatherReporting {2 6 6 247 1} DEFINITIONS IMPLICIT TAGS ::=
BEGIN
WeatherReport ::= SEQUENCE {
    height [0] REAL,
    weather [1] Wrecord
}
Wrecord ::= [APPLICATION 3] EXPLICIT SEQUENCE {
    temp Temperature,
    moist Moisture
    wspeed [0] EXPLICIT Windspeed OPTIONAL
}
Temperature ::= [APPLICATION 0] REAL
Moisture ::= [APPLICATION 1] EXPLICIT REAL
Windspeed ::= [APPLICATION 2] EXPLICIT REAL
END - - of module
WeaterhReporting
```

Exercise 22 Solution

Tags which can be removed are shown as italics below.

```
Colour ::= [APPLICATION 0] CHOICE {  
    rgb [1] RGB-Colour,  
    cmg [2] CMG-Colour,  
    freq [3] Frequency  
}
```

```
RGB-Colour ::= [APPLICATION 1] SEQUENCE {  
    red [0] REAL,  
    green [1] REAL OPTIONAL,  
    blue [2] REAL  
}
```

```
CMG-Colour ::= SET {  
    cyan [1] REAL,  
    magenta [2] REAL,  
    green [3] REAL  
}
```

```
Frequency ::= SET {  
    fullness [0] REAL,  
    freq [1] REAL  
}
```

Exercise 23a Solution

```
ListResult ::= OPTIONALLY-SIGNED  
  CHOICE {  
    listInfo SET {  
      DistinguishedName OPTIONAL,  
      subordinates [1] SET OF SEQUENCE {  
        RelativeDistinguishedName,  
        aliasEntry [0] BOOLEAN DEFAULT FALSE  
        fromEntry [1] BOOLEAN DEFAULT TRUE},  
      partialOutcomeQualifier [2]  
      PartialOutcomeQualifier OPTIONAL  
      COMPONENTS OF CommonResults },  
    uncorrelatedListInfo [0] SET OF Listresult }
```

Exercise 23b Solution

Yes, two comma characters are missing, see below

```
ListResult ::= OPTIONALLY-SIGNED
  CHOICE {
    listInfo SET {
      DistinguishedName OPTIONAL,
      subordinates [1] SET OF SEQUENCE {
        RelativeDistinguishedName,

        aliasEntry [0] BOOLEAN DEFAULT FALSE,
        fromEntry [1] BOOLEAN DEFAULT TRUE},
      partialOutcomeQualifier [2]

      PartialOutcomeQualifier OPTIONAL,
    COMPONENTS OF CommonResults },
    uncorrelatedListInfo [0] SET OF Listresult }
```

Exercise 23c Solution

Answer: **COMPONENTS OF** is not a data type, and can thus not have any identifier. It copies a series of separately defined type elements, and is useful if you have a series of standard elements, like **CommonResults**, which is to be used in many places.

Exercise 23d Solution

Answer: in a **SET** all the elements must have different type. It is then necessary to give a context tag only on all but one of the elements.

Exercise 24 Solution

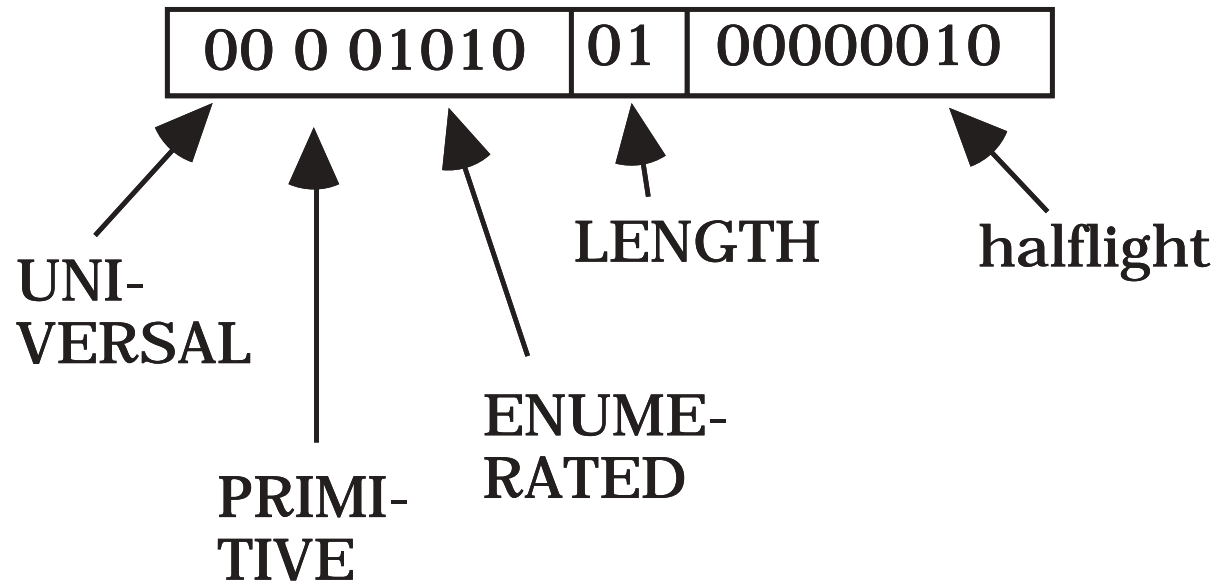
```
CarDriving { 1 2 4711 18 } DEFINITIONS EXPLICIT TAGS ::=
  BEGIN
    IMPORTS MainOperation FROM Driving {1 2 4711 17};
  FullOperation ::= SEQUENCE {
    COMPONENTS OF MainOperation,
    blink SEQUENCE {
      on BOOLEAN,
      left BOOLEAN },
    light ENUMERATED {
      dark(0),
      parkingLight (1),
      dimmedLight (2),
      fullBeam (3)
    } }
  END -- of module CarDriving
```

Note: Since there was no **EXPORTS** statement in **Driving**, all objects in it are exported.

Exercise 25 Solution

APPLI- CATION	CON- STRUC- TED	Tag nr.	Length	UNI- VER- SAL	PRIMI- TIVE	IA5- STRING	Lengt h				
01	1	00001	6	00	0	10110	4	M	a	r	y
61			06	16			04	M	a	r	y

Exercise 26 Solution



Exercise 27 Solution

(I make no promise that this is 100 % correct!)

element	encoding	Octets
beverage	(context explicit tag) 101 00001 (ENUMERATED) 000 01010	2
tea	(length) 1 (value) 00000001	2
jam	(context explicit tag) 101 00010 (ENUMERATED) 000 01010	2
orange	(length) 1 (value) 00000000	2

element	encoding	Octets
continentalpart	(SEQUENCE) 001 10000 (length) 8 beverage tea jam orange	10
eggform fried	(ENUMERATED) 000 01010 (length) 1 (value) 00000101	3
english	(SEQUENCE) 001 10000 (length) 10 continentalpart	12
typeofbreakfast	(context explicit tag) 100 00001 (length) 12 english	14
customername	(IA5string) 00010110 (length) 5 ("Johan") "J" "o" "h" "a" "n"	7

element	encoding	Octets
firstorder	(SEQUENCE) 001 10000 (length) 21 customername typeofbreakfast	23