ICM – Computer Science Major and M1 Cyber Physical and Social Systems
Data Interoperability and Semantics

Exam "PT1H30M"^^xsd:duration

You are allowed to use an A4 sheet of paper, hand-written, on both side.
You are not allowed to use any computer or electronic device during the test.

Part 1 (5 points) Comparison of main data formats (target: 20min for this exercise)

On one large full-page table, shortly describe or compare each data formats from List 1 in terms of the features and criteria from List 2.
NOTE: Be concise. You don't need to justify.

List 1. Data Formats: 1. CSV 2. JSON 3. XML 4. YAML

List 2: Features and criteria

1. **Simplicity:** How easy is the format to create, read, and modify by both humans and machines?
2. **Human-Readability:** Can the data be easily understood and edited in a text editor or similar tool?
3. **Interoperability:** Is the format widely supported across different tools, software, and platforms?
4. **File Size/Overhead:** How lightweight is the format in terms of file size, especially for large datasets?
5. **Data Type Support:** Does the format natively support a variety of data types (e.g., numbers, dates, booleans) and complex structures (e.g., arrays, objects)?
6. **Metadata Support:** Does the format allow embedding of metadata (e.g., column types, descriptions, units) within the file?
7. **Structural Complexity:** Can the format handle complex data structures, such as nested, hierarchical, relational, or graph data?
8. **Query:** Are there dedicated query language for the format?
9. **Schema Validation:** Does the format support a schema or mechanism to enforce data consistency and integrity?
10. **Extensibility:** Can the format be extended with custom structures or elements without breaking compatibility (e.g., custom tags or schemas)?

Part 2 (/20 points) ASN.1

ASN.1 is a standard *interface description language* for defining data structures that can be serialized and deserialized in a cross-platform way. It provides a formal way to describe data and enables interoperability across different systems by ensuring consistent data encoding and decoding. While ASN.1 is now 40 years old, it is still broadly used in telecommunication and computer networking, such as for 5G mobile phone communications, LDAP directories, Securing HTTP communications with TLS (X.509) Certificates, Intelligent Transport Systems, and the Interledger Protocol for digital payments.

Protocol developers define data structures in ASN.1 *modules*, which are generally a section of a broader standards document written in the ASN.1 language. Here are some common ASN.1 base data types.

BOOLEAN [tag number 01_{10}]: value can be TRUE or FALSE
INTEGER [tag: 02_{10}]: a signed integer. A valid range can be specified with the notation (min..max)
BIT STRING [tag number 03_{10}]: used for bit arrays, where each bit has an individual meaning.
ENUMERATED [tag number 10_{10}]: a list of named items.
SEQUENCE [tag number 16_{10}]: a collection of items to group together.
CHOICE [n/a]: one of the items can be present at a time.
IA5String [tag number 22_{10}]: a printable ASCII string.

Below is an ASN.1 module definition, adapted from the ETSI Intelligent Transport Systems (ITS) Common Data Dictionary definition https://forge.etsi.org/rep/ITS/asn1/cdd_ts102894_2.
*Note: "**ego**" is how we name the vehicle on which the communicating ITS system is deployed. "**alter**" is another vehicle that is observed by "ego", or that communicates with "ego".*

```
ETSI-ITS-CDD DEFINITIONS AUTOMATIC TAGS ::=
BEGIN
EgoData ::= SEQUENCE { -- invented for the exam -
    id                 IA5String,
    energyStorage      EnergyStorageType,
    speed              SpeedValue,
    driveDirection     DriveDirection,
    lanePosition       LanePositionOptions
}

AlterData ::= SEQUENCE { -- invented --
    id                 IA5String OPTIONAL,
    message            IA5String OPTIONAL,
    safeDistance       SafeDistanceIndicator,
    speed              SpeedValue,
    driveDirection     DriveDirection,
    lanePosition       LanePositionOptions
}

EnergyStorageType ::= BIT STRING { -- real def --
    hydrogenStorage        (0),
    electricEnergyStorage  (1),
    liquidPropaneGas       (2),
    compressedNaturalGas   (3),
    diesel                 (4),
    gasoline               (5),
    ammonia                (6)
}(SIZE(7))

SafeDistanceIndicator::= BOOLEAN -- real def --

SpeedValue ::= INTEGER { -- unit is 0,01 m/s --
    standstill  (0),
    outOfRange  (16382),
    unavailable (16383)
} (0..16383)

DriveDirection ::= ENUMERATED { -- real def --
    forward     (0),
    backward    (1),
    unavailable (2)
}

LanePositionOptions ::= CHOICE { -- real def --
    simplelanePosition       LanePosition,
    simpleLaneType           LaneType,
    detailedlanePosition     LanePositionAndType,
    ...
}

LanePositionAndType::= SEQUENCE { -- real def --
    transversalPosition LanePosition,
    laneType            LaneType DEFAULT traffic,
    ...
}

LanePosition ::= INTEGER { -- real def --
    offTheRoad          (-1),
    innerHardShoulder   (0),
    outerHardShoulder   (14)
} (-1..14)

LaneType::= INTEGER{ -- simplified: only some
values --
       traffic              (0),
       pedestrian           (12),
       parking              (17),
       emergency            (18)
       ...
}(0..31)
END
```

Because ASN.1 is both human-readable and machine-readable, an ASN.1 compiler can compile modules into libraries of code, *codecs*, that decode or encode the data structures.

ASN.1 defines different *encoding rules* that specify how to represent a data structure as bytes. Basic Encoding Rules (BER) is the oldest one, Packed Encoding Rules (PER) is the most compact. XML Encoding Rule (XER) is based on XML. JSON encoding rules (JER) is the easiest to start playing with ASN.1 and to debug applications.

In this part we will consider two messages, one about “ego”, one about “alter”:
Message 1: *“Ego is a Highway Grass Cutting Machine with id AB-123-CD. It uses and stores diesel, liquid propane gas, and electricity. It drives forward on the outer hard shoulder at a speed of 10.8 km/h.”*
Message 2: *“Alter EF-456-GH is moving forward at 144 km/h on traffic lane number 3, not respecting safe distances”*

Note: Check out the appendices A-E, as they are all important to answer the questions in this part.

Question 1. (1 pt) Justify that the encoded value for speed 10.8 km/h is integer 300.

Question 2. (2 pts)
- How IEEE 754 floating point number would encode the value 144.0 ?
- Find an IEEE 754 floating point number that approximates 10.8 at ±0.05

Question 3. (4 pts) Write a document that could be a plausible JER encoding of Message 1 about “ego”

Question 4. (1 pt) BER-encode the BIT STRING. `diesel+liquidPropaneGas+electricEnergyStorage`

The XML document below is the XER encoding of Message 2 about “alter”:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <AlterData>
3      <id>EF-456-GH</id>
4      <safeDistance>
5          <false />
6      </safeDistance>
7      <speed>4000<speed>
8      <driveDirection>
9          <forward />
10     </driveDirection>
11     <lanePosition>
12         <detailedlanePosition>
13             <transversalPosition>3</transversalPotion>
14             <laneType>traffic</laneType>
15         </detailedlanePosition>
16     </lanePosition>
17 </AlterData>
```

Question 5 (1 pt) I injected two syntax errors in this document. For each error, give the line number and explain it.

Question 6. (1 pt) What would be the BER encoding of (a) positive integer 4000 ? (b) negative integer -120 ?

Question 7. (1 pt) Justify that the maximal encodable length in BER is 2^{1008}

Question 8. (1 pt) Assume we want to set the `message` IA5String in Message 2 about “alter” as follows:
“Warning: Automated grass cutter ahead. Maintain safe distance. Speed reduced. Hazardous debris possible. Stay alert for sudden stops and avoid lane changes near the vehicle. Thank you for your cooperation.”
This message has a total of 205 characters. Give the most significant four bytes of that message encoded using BER

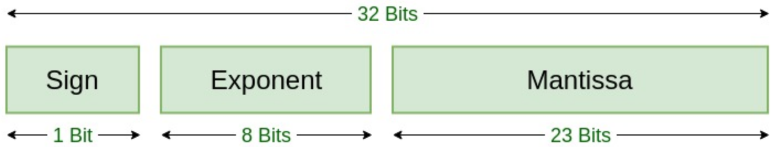
Question 9. (1 pt) Justify that the identifier octet for a SEQUENCE needs to be 30_{16}

Question 10. (4 pts) Determine the BER-encoding for Message 2 about “alter”. Justify step by step.

Question 11. (1 pt) What can go wrong with id and message being both OPTIONAL in the definition of `AlterData` ? Suggest additional encoding rules involving bits 8 and 7 of the identifier octet to avoid this issue.

Question 12 (2 pts) The most compact ASN.1 encoding rules are the Packed Encoding Rules (PER). This is the one commonly used in 3GPP cellular technologies such as UMTS (3G), LTE (4G), or 5G.
Give (4 maximum) concrete ideas for how PER greatly improves compaction with respect to BER.

Appendix A: Single precision IEEE 754 floating-point standard



$$-1^{s} \times 2^{(\exp-127)} \times 1.frac$$

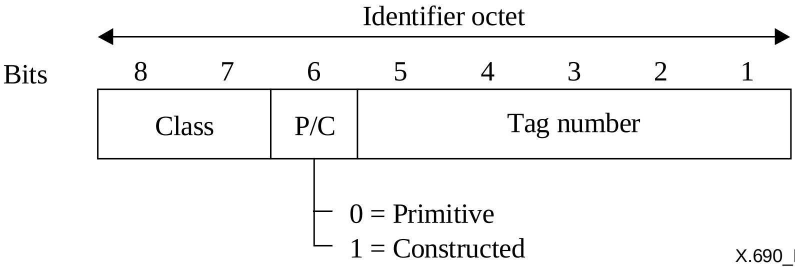
Appendix B: ASCII Table

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[END OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Appendix C: Introduction to ASN.1 Basic Encoding Rules

The Basic Encoding Rules (BER) uses a Tag-Length-Value (TLV) format for encoding all information. The tag indicates the **identifier** of the data that follows, the length indicates the total length of value (in bytes), and the value represents the actual data **contents**. Each value may consist of one or more TLV-encoded values, each with its own tag, length, and value.

Identifier octet (simplified)
The identifier octet encodes the ASN.1 tag of the type of the data value as follows:



X.690_F3

In the context of this exam, we assume that bits 8 and 7 are always set to 0.
bit 6 shall be a 0 if the data contents is primitive, or 1 if it is constructed.
bits 5 to 1 encode the number of the tag as a binary integer with bit 5 as the most significant bit.

Encoding Lengths
Length is always specified in octets, and includes only the octets of the actual value (the contents). It does not include the lengths of the identifier or of the length field itself. We consider two ways to encode lengths:

Short form: for lengths between 0 and 127, the one-octet short form can be used. In the encoding below, bit 8 of the length octet is set to 0, and the length is encoded as an unsigned binary value in the octet's rightmost seven bits. 8 1 [0 | Length]	**Long form**: for lengths between 0 and 2^{1008} octets, the long form can be used. It starts with an octet that contains the length of the length, followed by the actual length of the encoded value. For example, if the first octet of the length contains the value 4, the actual length of the contents is contained in the next four octets. 8 1 8 1 8 1 [1 | N | Length N | ... | Length 1]

Encoding of a boolean value: The encoding of a boolean value shall be primitive. The contents octets shall consist of a single octet. 0 if the value is FALSE, non-zero if the value is TRUE

Encoding of an integer value: The encoding of an integer value shall be primitive.
An integer value is encoded as a two’s complement binary number on the smallest possible number of octets.

Encoding of a bit string: See appendix D “BER encoding of a bitstring value”

Encoding of an enumerated value
The encoding of an enumerated value shall be that of the integer value with which it is associated.

Encoding of a sequence value: See appendix E “BER encoding of a sequence value”

Encoding of a choice value
The encoding of a choice value shall be the same as the encoding of a value of the chosen type.

NOTE 1 – The encoding may be primitive or constructed depending on the chosen type.
NOTE 2 – The tag used in the identifier octets is the tag of the chosen type, as specified in the ASN.1 definition of the choice type.

Appendix D: BER encoding of a bitstring value

This is an excerpt of Rec. ITU-T X690 (pp. 8-9), available online at https://www.itu.int/rec/T-REC-X.690

8.6 Encoding of a bitstring value

8.6.1 The encoding of a bitstring value shall be either primitive or constructed at the option of the sender.

NOTE – Where it is necessary to transfer part of a bit string before the entire bitstring is available, the constructed encoding is used.

8.6.2 The contents octets for the primitive encoding shall contain an initial octet followed by zero, one or more subsequent octets.

8.6.2.1 The bits in the bitstring value, commencing with the leading bit and proceeding to the trailing bit, shall be placed in bits 8 to 1 of the first subsequent octet, followed by bits 8 to 1 of the second subsequent octet, followed by bits 8 to 1 of each octet in turn, followed by as many bits as are needed of the final subsequent octet, commencing with bit 8.

NOTE – The terms "leading bit" and "trailing bit" are defined in Rec. ITU-T X.680 | ISO/IEC 8824-1 as follows: *The first bit in a bit string is called the leading bit. The final bit in a bit string is called the trailing bit.*

8.6.2.2 The initial octet shall encode, as an unsigned binary integer with bit 1 as the least significant bit, the number of unused bits in the final subsequent octet. The number shall be in the range zero to seven.

8.6.2.3 If the bitstring is empty, there shall be no subsequent octets, and the initial octet shall be zero.

[…]

8.6.4.2 Example

If of type **`BIT STRING`**, the value **`'0A3B5F291CD'H`** can be encoded as shown below. In this example, the bit string is represented as a primitive:

BitString	Length	Contents
03_{16}	07_{16}	$040A3B5F291CD0_{16}$

Appendix E: BER encoding of a sequence value

This is an excerpt of Rec. ITU-T X690 (pp. 10), available online at https://www.itu.int/rec/T-REC-X.690

8.9 Encoding of a sequence value

8.9.1 The encoding of a sequence value shall be constructed.

8.9.2 The contents octets shall consist of the complete encoding of one data value from each of the types listed in the ASN.1 definition of the sequence type, in the order of their appearance in the definition, unless the type was referenced with the keyword OPTIONAL or the keyword DEFAULT.

8.9.3 The encoding of a data value may, but need not, be present for a type which was referenced with the keyword OPTIONAL or the keyword DEFAULT. If present, it shall appear in the encoding at the point corresponding to the appearance of the type in the ASN.1 definition.

EXAMPLE

If of type:

```
SEQUENCE {name  IA5String,  ok  BOOLEAN}
```

the value:

```
{name  "Smith",  ok  TRUE}
```

can be encoded as:

Sequence	Length	Contents		
30_{16}	$0A_{16}$			
		IA5String	Length	Contents
		16_{16}	05_{16}	"Smith"
		Boolean	Length	Contents
		01_{16}	01_{16}	FF_{16}