A.1
a) It is utilized to isolate the information from the show.
b) XML is utilized to store the information.
c) It is utilized to structure the information.
d) It is likewise utilized for reloading of information bases.
e) It facilitates the production of HTML reports

A.2
<root>
<child>
<subchild> </subchild>
</child>
</root>
example :
<?xml version="1.0" encoding="UTF-8"?>
<school>
<name> Convent of Jesus and Mary </name>
<classes> Till twelve </classes>
<medium> English </medium>
<teachers> graduates </teachers>
</school>

A.3
<?xml version="1.0" encoding="UTF-8"?>
The previously mentioned is the XML prolog
It is a segment which is constantly composed toward the beginning of the XML report. It incorporates revelation, type, handling types and so forth.

A.4
e) None of these

A.5
date attribute inside message tag can be used as date tag inside message tag as follows:
<message>
<date>2020-01-22</date>
<to>Students</to>
<from>Teacher</from>
</message>
date tag is again expanded as

<message>
<date>
<year>2020</year>
<month>01</month>
<day>22</day>
</date>
<to>Students</to>
<from>Teacher</from>
</message>

A.6
CDATA stands for CHARACTER DATA
CDATA is used when you don't want to parse text or content. It means the content placed between CDATA is not not parsed by the parser.
SYNTAX
<![CDATA[
Characters won't be parsed here
]]>

A.7
XSL stands for Extensible Styleshet Language, its like CSS, it describes how to display an XML dcoument.

Example:-
XML:- sample XML to format
<cheggprofile>
<student name="student1">Welcome Student! Start asking questions</student>
<expert name="expert1">Welcome Expert! Start answering questions</expert>
</cheggprofile>

XSL:- following XSL code formats above XML, make text bold and background color "red" for expert profile

<xsl:template match="student">
<fo:block font-weight:"bold">
<xsl:apply-templates/>
</fo:block>
</xsl:template>

<xsl:template match="expert">
<fo:block font-weight:"bold" background-color="red">
<xsl:apply-templates/>
</fo:block>
</xsl:template>

A.8
a)- <xsl:for-each select="bookstore/book">
Ans:- this "xsl:for-each" tag, iterates through all sub elements <book> under <bookstore> and then you can apply transformation on each book:-
Example:-
XML:-
<bookstore>
<book><name>Game of Thrones</name><price>200$</price><year>2001</year></book>
<book><name>Harry Potter</name><price>50$</price><year>2010</year></book>
</bookstore>
XSL:- following xsl code iterates through the xml and prints book names and year published in the table
<table>
<tr>
<th>Title</th>
<th>Year</th>
</tr>

(b) <xsl:sort select="year">
Example:-
XML:-
<bookstore>
<book><name>Game of Thrones</name><price>200$</price><year>2001</year></book>
<book><name>Harry Potter</name><price>50$</price><year>2010</year></book>
</bookstore>
XSL:- following xsl code will sort the output table by year column
<table>
<tr>
<th>Title</th>
<th>Year</th>

```
</tr>
<xdl:for-each select="bookstore/book">
```

A.9 (i don't get the question but I try to answer that)
Parsing XML with Java Script

```
<script type="text/javascript">
    // get value of single node
    var descriptionNode = xmlData.getElementsByTagName("description")
[0];
var description
= descriptionNode.firstChild && descriptionNode.firstChild.nodeValue;
    // get values of nodes from a set
var relatedItems    = xmlData.getElementsByTagName("related_item");
// xmlData is an XML doc
        var relatedItemVals = [];
      var tempItemVal;
for (var i=0,total=relatedItems.length; i<total; i++){
  tempItemVal = relatedItems[i].firstChild ?
relatedItems[i].firstChild.nodeValue : "";
       relatedItemVals.push(tempItemVal);
}
// set and get attribute of a node
description.setAttribute("language", "en");
description.getAttribute("language"); // returns "en"
</script>
```

Look more closely at this code. The method getElementsByTagName(), which you saw before, is essential for processing XML because it allows you to select all XML elements of a given name. You then safely retrieve the description value by first checking if the descriptionNode has a firstChild. If so, you go on to access its nodeValue. When you try to access a specific node's text value, things start to get a little tricky. Although some browsers support the previously covered innerHTML property for XML documents, most do not. You first have to check whether it has a firstChild and if it does, retrieve that nodeValue. If the value doesn't exist, you set it to an empty string.
Last, you see that setAttribute() and getAttribute() methods work as they did with an HTML file.
Parsing XML with jQuery

The first argument passed to the jQuery $() function is the string selector. The less common second argument allows you to set the context, or starting node for jQuery, to use as a root when making the selection. By default, jQuery uses the document element as the context, but optimizing code is possible by restricting the context to a more specific (and therefore smaller) subset of the document. To process XML, you want to set the context to the root XML document .

```
<script type="text/javascript">
    // get value of single node (with jQuery)
    var description = $("description", xmlData).text();
        // xmlData was defined in previous section
    // get values of nodes from a set (with jQuery)
var relatedItems = $("related_item", xmlData);
var relatedItemVals = [];
$.each(relatedItems, function(i, curItem){
    relatedItemVals.push(curItem.text());
});
</script>
```

A.10
JSON

A.11
following line prints <<John Smith is 43>>
$("div").append(data.name + ' is ' + data.age)

A.12
you can use following getJSON function to read external json files
$.getJSON('external.json', function(json_data){console.log(json_data);});

A.13
following line will fetch <<Jane Doe>>, user[0] get first record from list and ".two" fetches "Jane Doe"
let result = data.users[0].two