Information Technology Final Exam

# Question 2

(a)

Line 1: exclamation point at end of xml tag, should be question mark

Line 3: 'NoOfAnimals' attribute not quoted.

Line 4: '<HabitatName>' tag doesn't have a closing tag. The '<Habitatname>' tag should be renamed to '</HabitatName>'

Line 5: '<HabitatManager>' tag doesn't have a closing tag. There should be a closing tag '</HabitatManager>'

added at line 9.

Line 9: '<HabitatManager>' tag doesn't have a closing tag. There should be a closing tag '</HabitatManager>'

added at line 14 (after previous changes, this becomes line 15).

Line 11: '<Collection\_Name>' doesn't have a closing tag. The '<Collection-Name>' tag should be renamed to '</Collection\_Name>'

Line 12: '<Status>' doesn't have a closing tag (as XML tags are case-sensitive). The '<status>' tag should be renamed to '</Status>'

Line 13: '</AnimalCollection> closing tag should not have an attribute attached to it. The attribute should go to the opening tag <AnimalCollection>.'

Line 17: '<Habitat>' does not have a closing tag. The '</Habtat>' tag should be renamed to '</Habitat>'.

|  |
| --- |
| <?xml version="1.0"?>  <Zoo>  <Habitat Code='AS' NoOfAnimals='2'>  <HabitatName>African Savanna</HabitatName>  <HabitatManager>  <title>Ms</title>  <firstname>Amy</firstname>  <lastname>Walsh</lastname>  </HabitatManager>  <HabitatManager>  <AnimalCollection CollectionID='10'>  <Collection\_Name>White Rhino</Collection\_Name>  <Status>Endangered</Status>  </AnimalCollection>  </HabitatManager>  </Habitat>  </Zoo> |

(b)

A valid XML document has a well-structured DTD that includes elements, attributes, entities, and cardinalities. This may or may not be included in the file itself. The XML should have exactly one root element, showing at least one element. Any non-empty tag must have a closing tag, and for any element, its given attribute (if it has one associated with it) must be quoted. All elements must be properly nested and indented.

(c)

|  |
| --- |
| <!DOCTYPE PharmacyCatalog [  <!ELEMENT PharmacyCatalog (Pharmacy+)>  <!ELEMENT Pharmacy (PharmacyName, PharmacyManager, Drug\*)>  <!ELEMENT PharmacyName (#PCDATA)>  <!ELEMENT PharmacyManager (title?, firstname+, lastname)>  <!ELEMENT Drug (Name, ActiveIngredient?)>  <!ELEMENT Name (#PCDATA)>  <!ELEMENT ActiveIngredient (#PCDATA)>  <!ELEMENT title (#PCDATA)>  <!ELEMENT firstname (#PCDATA)>  <!ELEMENT lastname (#PCDATA)>  <!ATTLIST Pharmacy PharmacyID CDATA #REQUIRED>  <!ATTLIST Pharmacy Category CDATA #IMPLIED>  <!ATTLIST Drug DrugID CDATA #REQUIRED>  <!ATTLIST Drug Controlled CDATA #REQUIRED>  ]> |

The XML file given showed three pharmacies. I assumed that a catalogue of pharmacies would have at least one pharmacy in it, as a catalogue with no items seems counterintuitive.

Each pharmacy had varying amounts of drugs, from no drugs at all to two types drugs. Due to this, I gave the ‘Drug’ sub element the “zero or more” type declaration. Each pharmacy also had one name and manager.

The manager either did or did not have a title, or had many or one first names. Due to this, I gave the ‘title’ sub element the “zero or one” type declaration and the ‘firstname’ sub element the “one or more” type declaration. Every manager also only had one last name.

Each ‘Drug’ element a name but may not have had an ‘ActiveIngredient’ sub element. Due to this, I gave the ‘ActiveIngredient’ sub element the “zero or one” type declaration.

Each pharmacy and drug had an ID associated with it, I gave these respective IDs the ‘#REQUIRED attribute type.

Only some pharmacies had a ‘Category’ attribute, so I also gave the pharmacy element an optional ‘Category’ attribute with the ‘#IMPLIED’ attribute type.

Lastly, every drug either was or was not controlled, determined by the value of its ‘Controlled’ attribute. But since each drug still had the attribute regardless of its value, I gave this attribute the ‘#REQUIRED’ attribute type.

(d)

(i)

Code:

|  |
| --- |
| for $p in doc("q2\_c.xml")  return fn:string-join($p//PharmacyName, '+') |

Expected results:

|  |
| --- |
| Greenhills Pharmacy+Beaumont Pharmacy+Jones Pharmacy |

This XQuery looks through every pharmacy and gets its ‘PharmacyName’ element. Then, using the built-in function ‘string-join’, I join the names with the ‘+’ character.

(ii)

Code:

|  |
| --- |
| for $c in doc("q2\_c.xml")//Pharmacy/@Category  return  <PharmacyCategories>  {$c}  </PharmacyCategories> |

Expected results:

|  |
| --- |
| <PharmacyCategories Category="community"/>  <PharmacyCategories Category="Hospital"/> |

This XQuery looks through every pharmacy and gets its ‘Category’ attribute. It then returns this attribute under the element ‘PharmacyCategories’. This code also ignores any pharmacies that don’t have the ‘Category’ attribute, so it doesn’t return a null tag, such as “<PharmacyCategories />”.

(iii)

Code:

|  |
| --- |
| (for $i in doc("q2\_c.xml")//PharmacyManager return ($i/firstname)[1]) |

Expected results:

|  |
| --- |
| <firstname>Helen</firstname>  <firstname>Frank</firstname>  <firstname>Annie</firstname> |

This XQuery looks through every pharmacy’s manager and returns the first firstname of each manager. It works by first getting each manager and returning their first names as a list. Then, using indexing, the code finally returns the first of the first names it comes across for each manager.

(iv)

Code:

|  |
| --- |
| for $p in doc("q2\_c.xml")//Pharmacy  where $p/@PharmacyID = 2  return  <PharmacyDetails>  {$p/PharmacyName}  {$p/PharmacyManager}  </PharmacyDetails> |

Expected results:

|  |
| --- |
| <PharmacyDetails>  <PharmacyName>Beaumont Pharmacy</PharmacyName>  <PharmacyManager>  <title>Mr</title>  <firstname>Frank</firstname>  <firstname>Brian</firstname>  <lastname>Dunne</lastname>  </PharmacyManager>  </PharmacyDetails> |

This XQuery looks through every pharmacy, using the ‘where’ expression, where the pharmacy has an ID equal to 2. It then returns that pharmacy’s name. Next, the code gets and returns the details about the pharmacy’s manager. Finally, all of this is wrapped inside an element called ‘PharmacyDetails’.