Tribhuvan University
Institute of Science and Technology
2079

Bachelor Level / Fifth Semester / Science
Computer Science and Information Technology (CSC318)
Web Technology
Full Marks: 60 + 20 + 20 Pass Marks: 24 + 8 + 8 Time: 3 Hours

Candidates are required to give their answers in their own words as far as practicable.

The figures in the margin indicate full marks.

Group A
Answer any two questions

What is the use of XML? Create an XML file with simple type and complex type elements. Write its equivalent DTD.

XML is a software- and hardware-independent tool for storing and transporting data.

What is XML?

- XML stands for eXtensible Markup Language
- XML is a markup language much like HTML
- XML was designed to store and transport data
- XML was designed to be self-descriptive
- XML is a W3C Recommendation

The Difference Between XML and HTML

XML and HTML were designed with different goals:

- XML was designed to carry data - with a focus on what data is
- HTML was designed to display data - with a focus on how data looks
- XML tags are not predefined like HTML tags are

XML Schema Definition commonly known as XSD is a way to describe precisely the XML language. XSDs check the validity of the structure and vocabulary of an XML document against the grammatical rules of the appropriate XML language.

```
<?xml version="1.0" encoding="UTF-8"?>
<Person>
```

```
  <Age>15</Age>
  <Name>Ram</Name>
  <Pincode>123</Pincode>
</Person>

<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

 <xs:element name="Person">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="Age" type="xs:int">
     <xs:annotation>
      <xs:documentation>Age must be between 10 and 20</xs:documentation>
     </xs:annotation>
     <xs:restriction base="xs:int">
      <xs:minInclusive value="10"/>
      <xs:maxInclusive value="20"/>
     </xs:restriction>
    </xs:element>
    <xs:element name="Name">
     <xs:annotation>
      <xs:documentation>Name must be either Ram or Shyam</xs:documentation>
     </xs:annotation>
     <xs:simpleType>
      <xs:restriction base="xs:string">
       <xs:enumeration value="Ram"/>
       <xs:enumeration value="Shyam"/>
      </xs:restriction>
     </xs:simpleType>
    </xs:element>
    <xs:element name="Pincode" type="xs:string">
     <xs:annotation>
      <xs:documentation>Pincode must be exactly 3 digits</xs:documentation>
     </xs:annotation>
     <xs:restriction base="xs:string">
      <xs:pattern value="\d{3}"/>
```

```
      </xs:restriction>
    </xs:element>
  </xs:sequence>
 </xs:complexType>
</xs:element>

</xs:schema>
```

XML (eXtensible Markup Language) is a flexible, structured format used to store and transport data. It's often used for:

1. **Data Interchange**: Between systems, applications, or organizations.
2. **Configuration Files**: For software settings.
3. **Data Storage**: For structured data in a readable format.
4. **Document Representation**: Such as in RSS feeds or SOAP messages.

Example XML File

Here's an XML file with both simple and complex type elements:

xml

Copy code

```xml
<?xml version="1.0" encoding="UTF-8"?>
<library>
    <book>
        <title>Introduction to XML</title>
        <author>Jane Doe</author>
        <year>2023</year>
        <price>29.99</price>
    </book>
    <book>
        <title>Advanced XML Techniques</title>
        <author>John Smith</author>
        <year>2021</year>
        <price>39.95</price>
    </book>
```

```
</library>
```

Equivalent DTD (Document Type Definition)

Here's a DTD that defines the structure of the above XML file:

dtd
Copy code

```
<!DOCTYPE library [
    <!ELEMENT library (book+)>
    <!ELEMENT book (title, author, year, price)>
    <!ELEMENT title (#PCDATA)>
    <!ELEMENT author (#PCDATA)>
    <!ELEMENT year (#PCDATA)>
    <!ELEMENT price (#PCDATA)>
]>
```

Explanation:

- **XML File**:
  - `<library>`: Root element containing multiple `<book>` elements.
  - `<book>`: A complex type element containing `<title>`, `<author>`, `<year>`, and `<price>` elements.
  - `<title>`, `<author>`, `<year>`, `<price>`: Simple type elements containing parsed character data (text).
- **DTD**:
  - `<!DOCTYPE library [...]>`: Declares the document type and includes the DTD.
  - `<!ELEMENT library (book+)>`: Defines that the `<library>` element must contain one or more `<book>` elements.
  - `<!ELEMENT book (title, author, year, price)>`: Defines that each `<book>` element must contain exactly one `<title>`, `<author>`, `<year>`, and `<price>`.

- `<!ELEMENT title (#PCDATA)>`, `<!ELEMENT author (#PCDATA)>`, `<!ELEMENT year (#PCDATA)>`, `<!ELEMENT price (#PCDATA)>`: Defines that these elements contain parsed character data.

This DTD ensures that the XML structure adheres to the specified format and constraints.

Create an HTML file containing form elements textbox for username, password field, and a checkbox for hobbies. Now write a JavaScript function for the form validation. You should validate for the username to be empty, the password should be of length at least 5, and the checkbox should be checked.

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width,
initial-scale=1.0">

    <title>Form Validation</title>

    <style>

        body {

            font-family: Arial, sans-serif;

        }

        form {

            max-width: 300px;

            margin: auto;

            padding: 20px;

            border: 1px solid #ddd;

            border-radius: 8px;

        }
```

```
        .error {
            color: red;
            font-size: 0.875em;
            margin-top: 5px;
        }
        .hidden {
            display: none;
        }
    </style>
</head>
<body>
    <h1>Signup Form</h1>
    <form id="signupForm" onsubmit="return validateForm()">
        <label for="username">Username:</label>
        <input type="text" id="username" name="username">
        <div id="usernameError" class="error hidden">Username cannot
be empty.</div>

        <label for="password">Password:</label>
        <input type="password" id="password" name="password">
        <div id="passwordError" class="error hidden">Password must
be at least 5 characters long.</div>

        <label for="hobbies">Hobbies:</label>
        <input type="checkbox" id="hobbies" name="hobbies">
```

```
        <div id="hobbiesError" class="error hidden">You must check
the hobbies checkbox.</div>


        <button type="submit">Submit</button>

    </form>


    <script>

        function validateForm() {

            let isValid = true;


            // Username Validation

            const username =
document.getElementById('username').value;

            if (username.trim() === '') {


document.getElementById('usernameError').classList.remove('hidden');

                isValid = false;

            } else {


document.getElementById('usernameError').classList.add('hidden');

            }


            // Password Validation

            const password =
document.getElementById('password').value;

            if (password.length < 5) {
```

```
document.getElementById('passwordError').classList.remove('hidden');
                isValid = false;
            } else {
document.getElementById('passwordError').classList.add('hidden');
            }

            // Hobbies Checkbox Validation
            const hobbiesChecked =
document.getElementById('hobbies').checked;
            if (!hobbiesChecked) {
document.getElementById('hobbiesError').classList.remove('hidden');
                isValid = false;
            } else {
document.getElementById('hobbiesError').classList.add('hidden');
            }

            // Return false if validation fails, preventing form
submission
            return isValid;
        }
    </script>
</body>
</html>
```

Explanation:

- **HTML Form**:
  - `username`: A textbox for entering the username.
  - `password`: A password field for entering the password.
  - `hobbies`: A checkbox for hobbies.
  - `onsubmit="return validateForm()"`: Calls the `validateForm` function when the form is submitted.
- **JavaScript Validation** (`validateForm` function):
  - Checks if the username is empty and shows an error if it is.
  - Ensures the password length is at least 5 characters and shows an error if it is not.
  - Verifies if the hobbies checkbox is checked and shows an error if it is not.
  - Returns `false` if any validation fails to prevent form submission; otherwise, it returns `true`.

How is a database connection created in PHP? Write a PHP program to create a form and insert values into the database table. The form should contain at least two input fields.

To create a database connection in PHP and create a form to insert values into a

database table, you'll need to follow these steps:

1. **Create a Database and Table:**

Before writing PHP code, make sure you have a database set up. Create a table to

store the form data. For example, let's assume you have a database named "example"

and a table named "users" with columns "id," "name," and "email."

```sql

CREATE DATABASE example;

USE example;

CREATE TABLE users (

id INT AUTO_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

email VARCHAR(255) NOT NULL

);
```

```

2. **PHP Code for Database Connection and Form:**

Now, create a PHP file (e.g., `index.php`) with the following code:

```php
<?php
// Database configuration
$servername = "your_database_host";
$username = "your_database_username";
$password = "your_database_password";
$dbname = "example";
// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
die("Connection failed: " . $conn->connect_error);
}
// Check if the form is submitted
if ($_SERVER["REQUEST_METHOD"] == "POST") {

// Get form data
$name = $_POST["name"];
$email = $_POST["email"];
// Insert data into the database
$sql = "INSERT INTO users (name, email) VALUES ('$name', '$email')";
if ($conn->query($sql) === TRUE) {
echo "Record inserted successfully!";
```
```

```
} else {
echo "Error: " . $sql . "<br>" . $conn->error;
}
}
// Close the database connection
$conn->close();
?>
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>PHP Form</title>
</head>
<body>
<h2>Insert Data into Database</h2>
<form method="post" action="<?php echo $_SERVER["PHP_SELF"]; ?>">
<label for="name">Name:</label>
<input type="text" name="name" required><br>
<label for="email">Email:</label>
<input type="email" name="email" required><br>
<input type="submit" value="Submit">
</form>
</body>
</html>
```

Make sure to replace "your_database_host," "your_database_username," and "your_database_password" with your actual database credentials.

3. **Run the Application:**

Place the PHP file (`index.php`) in your web server's document root and access it through a web browser. Fill out the form

Group B

Answer any eight questions

1. Differentiate Web 1.0 from Web 2.0.

Aspect	Web 1.0	Web 2.0
Nature	Static Content	Dynamic Content
User Interaction	Read-Only	Read-Write
Design and Functionality	Basic, simple designs	Rich, interactive designs
Content Management	Manual updates	Content Management Systems (CMS)
Technologies	HTML, basic CSS, limited JavaScript	HTML5, CSS3, JavaScript frameworks (AJAX)
User Engagement	Minimal, passive consumption	Active, collaborative, user-generated content
Examples	Early websites, static company pages	Social media, blogs, wikis, content-sharing platforms
Interactivity	Low	High
Data Handling	Server-side, static files	Client-side scripting, dynamic updates
Content Creation	Admins only	Users contribute and create content
Technological Advancements	Limited to basic HTML/CSS	Advanced scripting, responsive design

Web 1.0 vs. Web 2.0

Feature	Web 1.0	Web 2.0
Nature	Read-only	Read-write
Content	Static	Dynamic
User Interaction	Limited	High
Focus	Information dissemination	User participation and collaboration
Examples	Early websites, online encyclopedias	Social media, blogs, wikis

2. Write an HTML script to generate the following list of items:
    - 1. Coffee:
        - I. Black coffee
        - II. Green coffee
    - 2. Tea
        - A: Black tea
        - B. Green tea

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width,
initial-scale=1.0">

    <title>List Example</title>

    <style>

        ul {

            list-style-type: none;

        }

    </style>

</head>

<body>

    <h1>List of Items</h1>

    <ol type="1">

        <li>Coffee:

            <ul>

                <li>I. Black coffee</li>

                <li>II. Green coffee</li>

            </ul>

        </li>

        <li>Tea

            <ol type="A">

                <li>Black tea</li>
```

```
                <li>Green tea</li>
            </ol>
        </li>
    </ol>
</body>
</html>
```

Explanation:

3. **`<ol type="1">`**: Creates an ordered list with numbers (1, 2, 3, etc.).
    - **`<li>Coffee:`**: The first list item.
    - **`<ul>`**: Creates a nested unordered list.
        - **`<li>`**: List items for "Coffee" with Roman numerals (I, II).
4. **`<li>Tea`**: The second list item.
    - **`<ol type="A">`**: Creates a nested ordered list with letters (A, B, C, etc.).
        - **`<li>`**: List items for "Tea" with letters (A, B).

5. Create an HTML page containing a div having id “dv1”. The div should contain a canvas element. The id of the canvas should be “mycanvas”. The height and width of the canvas should be 200 and 300.

Web 1.0 vs. Web 2.0

The internet has evolved significantly since its inception. This evolution can be categorized into distinct phases: Web 1.0 and Web 2.0.

Web 1.0

Often referred to as the "Read-Only Web," Web 1.0 was primarily about consuming information. Websites were static, with limited interaction and user-generated content. Key characteristics include:

- **Static content:** Information was primarily text-based and rarely updated.
- **Limited interaction:** Users could mainly read information but couldn't contribute.
- **Focus on information:** Websites were primarily used as online brochures.

- **Examples:** Early websites, online encyclopedias like Britannica.

Web 2.0

Known as the "Read-Write Web," Web 2.0 emphasized user participation and interaction. It marked a shift from passive consumption to active creation and sharing. Key characteristics include:

- **Dynamic content:** Websites frequently updated with new content.
- **User-generated content:** Users can create and share content (blogs, videos, images).
- **Social interaction:** Platforms for social networking and collaboration emerged.
- **Rich media:** Increased use of images, videos, and audio.
- **Examples:** Social media platforms (Facebook, Twitter), video-sharing sites (YouTube), blogs, and wikis.

In essence:

- Web 1.0 was about accessing information.
- Web 2.0 is about creating, sharing, and interacting with information.

The transition from Web 1.0 to Web 2.0 has fundamentally changed how we interact with the Internet. Web 2.0 has empowered users to become creators and contributors, fostering a more dynamic and interconnected online world.

6. What are the usages of class and id selectors in CSS? Illustrate with an example.

Class and ID Selectors in CSS

CSS selectors are used to target specific HTML elements to which you want to apply styles. Among these, class and ID selectors are the most commonly used.

ID Selector

- **Purpose:** Used to style a unique element on a page.
- **Syntax:** `#id_name`

- **Usage:**
    - When you need to style a single element that won't be reused.
    - For accessibility purposes as it can be used as an anchor for jumping to a specific section of a page.

```
<div id="heading">This is a heading</div>

#heading {
 color: red;
 font-size: 24px;
}
```

Class Selector

- **Purpose:** Used to style multiple elements with the same style.
- **Syntax:** `.class_name`
- **Usage:**
    - When you want to apply the same styles to multiple elements.
    - For creating reusable style groups.

Example:

```
<p class="highlight">This is a highlighted paragraph.</p>

<div class="highlight">This is a highlighted div.</div>

.highlight {
 background-color: yellow;
 font-weight: bold;
}
```

Feature	ID Selector	Class Selector
Uniqueness	Unique per page	Can be used multiple times
Syntax	`#id_name`	`.class_name`
Use cases	Single element styling, accessibility	Multiple element styling, reusability

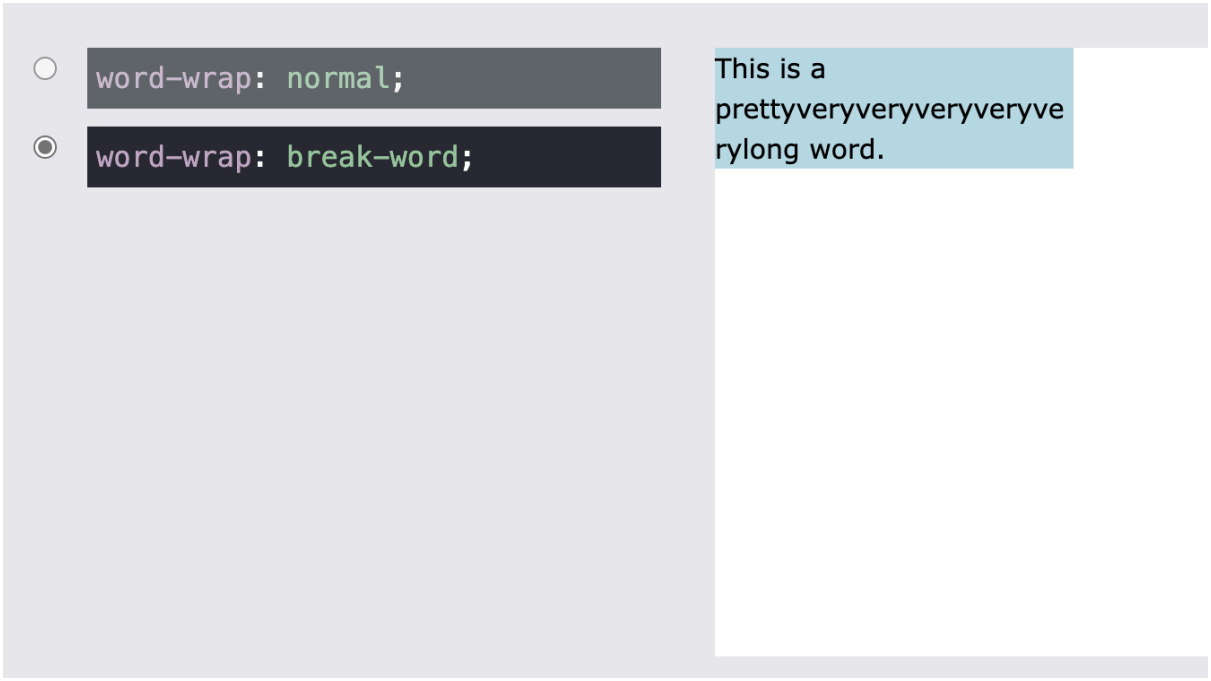
In Summary:

- Use an ID selector when you want to style a single, unique element.
- Use a class selector when you want to apply the same style to multiple elements.

By effectively using class and ID selectors, you can create well-structured and maintainable CSS stylesheets.

7. What is the word-wrap property in CSS? Write an HTML script to illustrate the word-wrap property.

The word-wrap property allows long words to be able to be broken and wrapped onto the next line.



normal Break words only at allowed break points. This is default

```
div {
  word-wrap: break-word;
}
```

Describe the jQuery Animate() method. Write an html script to create a div with id “mydiv”. The position of the div should be set to absolute. Now show the use of animate method to the div element to right by 300px.

```
<!DOCTYPE html>
<html>
<head>
 <title>jQuery Animate Example</title>
 <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
</head>
<body>
 <div id="myDiv">This is a div</div>

 <script>
  $(document).ready(function() {
    $("#myDiv").css({
      position: "absolute",
      left: 0
    });

    $("#myDiv").animate({ left: "+=300px" }, 1000);
  });
 </script>
```

```
</body>

</html>
```

jQuery Animate() Method

The jQuery animate() method is used to create custom animations for HTML elements. It allows you to gradually change CSS properties over a specified duration

Explanation:

1. **Include jQuery:** The <script> tag links the jQuery library.
2. **Create a div:** A div with the ID "myDiv" is created.
3. **Set initial position:** The $(document).ready() function ensures the script runs after the DOM is loaded. Inside this function, the css() method sets the position of the div to absolute and its initial left position to 0.
4. **Animate the div:** The animate() method is used to create the animation.

- The first argument is an object specifying the CSS properties to be changed. Here, we're changing the left property. The +=300px syntax increments the current left value by 300 pixels.
- The second argument is the duration of the animation in milliseconds. In this case, it's 1000 milliseconds (1 second).

When you run this code, you'll see the div positioned absolutely at the top-left corner of the page and then smoothly move 300 pixels to the right over one second.

Note: You can animate multiple properties simultaneously by passing an object with multiple properties to the animate() method.

What is the XML namespace? how it is used to avoid element name conflict in XML? justify with an example.

XML namespaces are used to avoid naming conflicts in XML documents. They allow you

to use elements and attributes with the same names but defined in different namespaces without causing conflicts. This is particularly useful when integrating XML from different sources or when different XML vocabularies are combined.

Here's an example to illustrate the need for XML namespaces and how they are used to avoid element name conflicts:

Suppose we have two XML documents representing information about books from different sources and they both use the same element names:

```
<book>
  <title>The Lord of the Rings</title>
  <author>J.R.R. Tolkien</author>
</book>
<person>
  <title>Dr. John Doe</title>
  <name>John Doe</name>
  <address>123 Main St</address>
</person>
```

Here, the element "title" has two different meanings.

```
<bookstore xmlns:bk="http://www.example.com/book">
  <bk:book>
    <bk:title>The Lord of the Rings</title>
      <bk:author>J.R.R. Tolkien</author>
```

```
  </bk:book>
</bookstore>
<person xmlns:p="http://www.example.com/person">
  <p:title>Dr. John Doe</p>
    <p:name>John Doe</p>
      <p:address>123 Main St</address>
</person>
```

In this example:

- xmlns:bk="http://www.example.com/book" declares a namespace with prefix "bk" and URI "http://www.example.com/book".
- xmlns:p="http://www.example.com/person" declares another namespace with prefix "p" and URI
- "http://www.example.com/person".
- The elements bk:title and p:title clearly differentiate between the two meanings of "title".

By using namespaces, we can effectively combine different XML vocabularies without causing conflicts, ensuring clarity and accuracy in the data.

How can you define classes and objects in PHP? Write a PHP script to create a class and its object.

Define a Class

A class is defined by using the class keyword, followed by the name of the class and a pair of curly braces ({}). All its properties and methods go inside the braces:

```
<?php
class Fruit {
```

```
  // code goes here...
}
?>
<?php
class Fruit {
  // Properties
  public $name;
  public $color;
  // Methods
  function set_name($name) {
    $this->name = $name;
  }
  function get_name() {
    return $this->name;
  }
}
?>
```

Note: In a class, variables are called properties and functions are called methods!

Define Objects

Classes are nothing without objects! We can create multiple objects from a class. Each object has all the properties and methods defined in the class but will have different property values.

Objects of a class are created using the new keyword.

In the example below, $apple and $banana are instances of the class Fruit:

```php
<?php
class Fruit {
  // Properties
  public $name;
  public $color;
  // Methods
  function set_name($name) {
    $this->name = $name;
  }
  function get_name() {
    return $this->name;
  } }
$apple = new Fruit();
$banana = new Fruit();
$apple->set_name('Apple');
$banana->set_name('Banana');
echo $apple->get_name();
echo "<br>";
echo $banana->get_name();
?>
```

How can you define an array in PHP? Write a PHP function to create an

an array of type integers and print its elements.

Defining an Array

In PHP, an array can be defined using the `array()` function or the shorter array syntax `[]`. For an integer array, you can simply list the integer values within the array.

```
// Using array() function

$integerArray = array(1, 2, 3, 4, 5);

// Using short array syntax

$integerArray = [1, 2, 3, 4, 5];
```

Creating a Function to Print Array Elements

Here's a PHP function to print the elements of an integer array:

```
<?php

function printIntArray($array) {

  foreach ($array as $value) {

    echo $value . " ";

  }

  echo "\n";

}

 $numbers = [10, 20, 30, 40, 50];

printIntArray($numbers);

?>
```

Explanation

- The `printIntArray` function takes an array as input.
- The `foreach` loop iterates over each element in the array.
- Inside the loop, the current element `$value` is printed followed by a space.
- After printing all elements, a newline character is printed.

This code defines an integer array and prints its elements using a dedicated function.