CMR INSTITUTE OF TECHNOLOGY

Internal Assessment Test 2 – November. 2017

Scheme and Solution

Sub:	Programming the Web							**Code:**	10CS73
Date**:**	08/ 11/2017	Duration:	90 mins	Max Marks:	50	**Sem:**	VII	**Branch:**	ISE

Note: Answer any five questions:

1. 6 M

a) Perl has three categories of variables/values: scalars, arrays, hashes

- Variables for each category are distinguished by the first symbol in the variable name
  - $ for scalar
  - @ for array
  - % for hash

1)**Scalars** come in three kinds: numbers, character strings and references

- Numbers are stored internally as double-precision floating point
- Note that strings are considered scalars in Perl

Scalar variable names begin with $, followed by letters digits and/or underscores

- Case sensitive
- Conventionally, programmer defined names do not use upper case letters

Scalar variable values are interpolated into double quoted strings

- If $x has the value 3
- Then "Value of x is $x" becomes "Value of x is 3"

Unassigned variables have the value undef

- undef converts to 0 as a number and the null string as a string

Perl has a large number of predefined variables

Many are named with special characters, such as $_ and $^

2) An **array** is a variable that stores a list

The name of an array variable begins with the character @

An array variable may be assigned a literal list value

- @a = (1, 2, 'three', 'iv');

An array assignment creates a new array as a copy of the original

- @b = @a;

3) **Hash variables** are named beginning with the character %

If an array is assigned to a hash, the even index elements become keys and the odd index elements are the corresponding values

- Assigning an odd length array to a hash causes an error

Curly braces are used to 'subscript' a hash

- If %h is a hash, then the element corresponding to 'four' is referenced as $h{'four'}

Values can be assigned to a hash reference to insert a new key/value relation or to change the value related to a key

A key/value relation can be removed from a hash with the delete operator

The undef operator will delete all the contents of a hash

The exists operator checks if a key is related to any value in a hash

- Just check $h{'something'} doesn't work since the related value may be the empty string or 0, both of which count as boolean false

	• A hash variable embedded in a string is not interpolated
	b)
	%countries=(India=>"Delhi", USA=>"Washington");
	foreach $capital(keys %countries)
	{
	print "capital is" $countries{$capital} \n";
	}	

2.	Explain object creation and modification in JavaScript with example.
	JavaScript Objects
	• **Objects are collections of *properties***
	• **Properties are either *data properties* or *method properties***
	• **Data properties are either primitive values or references to other objects**
	• **Primitive values are often implemented directly in hardware**
	• **The Object object is the ancestor of all objects in a JavaScript program**
	• **Object has no data properties, but several method properties**
	The usual way to create any object is with the new operator and a call to a constructor. In the case of arrays, the constructor is named Array:
	`var my_list = new Array(1, 2, "three", "four");`
	`var your_list = new Array(100);`
	The second way to create an Array object is with a literal array value, which is a list of values enclosed in brackets:
	var my_list_2 = [1, 2, "three", "four"];
	Object Modification
	To add a new property to an object, specify the object name followed by: a dot, the name of the new property, an equals sign, and the value for the new property (enclosed in quotes if it is a string).
	myObject.prop2 = 'data here';
	To change the value of an existing property of an object, specify the object name followed by: a dot, the name of the property you wish to change, an equals sign, and the new value you wish to assign.
	myObject.sProp = 'A new string value for our original string property';
	An expression with an object name, a dot, and a property name (myObject.sProp) will evaluate to the current value of that property. Our example first displays the value in an alert, then assigns the value to the variable val.
	alert(myObject.sProp) // display myObject.sProp value in alert
	var val = myObject.sProp;
		10M

3.	Write a Perl program to read an array of names from a file sort them and write in reverse order in another file.
	$index=0;
	open(FH1, "input.txt");
	open(FH2, "output.txt");
	while($content=<FH1>)
		8 M

```
{
  $names[$index++]=chomp($content)

}
@sorted_names=sort @names;

close(FH1);

foreach $name(reverse @names)
{
print FH1 $name "\n";
}
close(FH2);
```

b) List out the string functions available in Perl. (2M)

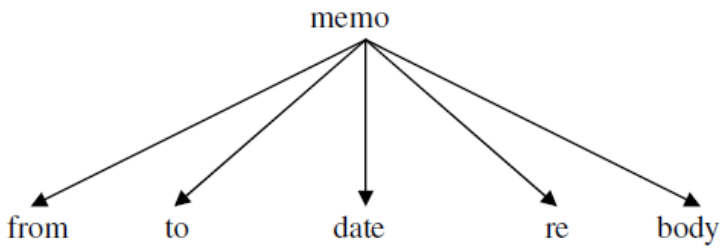
Name	Actions
Chomp	Removes any terminating newline characters* from its parameter; returns the number of removed characters
length	Returns the number of characters in its parameter string
lc	Returns its parameter string with all uppercase letters converted to lowercase
uc	Returns its parameter string with all lowercase letters converted to uppercase
hex	Returns the decimal value of the hexadecimal number in its parameter string
join	Returns a string constructed by catenating the strings of the second and subsequent strings together, with the parameter character inserted between them

4. Write a XHTML code to read Name, Email-id and Phone number and write a JavaScript to validate the three fields (name must have first name and last name with first character in capital, phone number must be of the form +91 XXXXXXXXXX(ten digits) ,and an valid Email- id). 10 M

```
<html>
<head><title> USN Laidation </title>
</head>
<script type="text/javascript">
 function validate(name,phone,email)

{
   <!-- Regular Expressing for  name, phone number and email-id-->

var name=/^[A-Z][a-zA-Z]*[\s][a-zA-Z]+$/;
var phoneno = /^[\+][91]\d{10}$/;
var mail_id=/^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/

 <!-- Checking USN Value with pattren  or USN length is zero -->
```

```
 if (!name.value.match(p1)||name.value.length==0)

 {
          alert(" Enter a valid name")

       return false;
 }

else
 if(!phone.value.match(p1)||phone.value.length==0)
       {

 alert("Enter a valid phone number")

    return false;


}
else
 if(!email.value.match(p1)||email.value.length==0)
       {

 alert("Enter a valid email-id")

    return false;

 }


else

{

   alert("All details are  valid") ;
return true;
}


 }
</script>



<body bgcolor="green" text="blue">

  <h1>  Validation</h1>

    <form method="post" action="">

Name: <input type="text" name="usn" value=""><br>

Phone: <input type="text" name="phone" value=""><br>

Email-ID: <input type="text" name="email" value=""><br>

      <input type="submit" value="Check" onclick="validate(name,phone,email) "><br>

     </form>
```

10 M

	</body> </html>	
5.	Explain three types that can be used to describe data in an element declaration with an example each. 1. **Declaring Elements:** • Element declarations are similar to BNF(CFG)(used to define syntactic structure of Programming language) here DTD describes syntactic structure of particular set of doc so its rules are similar to BNF. • An element declaration specifies the name of an element and its structure • If the element is a leaf node of the document tree, its structure is in terms of characters • If it is an internal node, its structure is a list of children elements (either leaf or internal nodes) • General form: <!ELEMENT element_name(list of child names)> e.g., <!ELEMENT memo (from, to, date, re, body)> This element structure can describe the document tree structure shown below.  Child elements can have modifiers, + -> One or more occurrences * -> Zero or more occurrences ? ->Zero or one occurrences Ex: consider below DTD declaration <!ELEMENT person (parent+, age, spouse?, sibling*)> One or more parent elements One age element Possible a spouse element. Zero or more sibling element. • Leaf nodes specify data types of content of their parent nodes which are elements 1. PCDATA (parsable character data) 2. EMPTY (no content) 3. ANY (can have any content) 2. **Declaring Attributes:** • Attributes are declared separately from the element declarations • General form: <!ATTLIST element_name attribute_name attribute_type [default _value]> More than one attribute < !ATTLIST element_name attribute_name1 attribute_type default_value_1 attribute_name 2 attribute_type default_value_2 …> Attribute type :There are ten different types, but we will consider only CDATA Possible Default value for attributes: Value - value ,which is used if none is specified #Fixed value - value ,which every element have and can't be changed # Required - no default value is given ,every instance must specify a value #Implied - no default value is given ,the value may or may not be specified Example : <!ATTLIST car doors CDATA "4"> <!ATTLIST car engine_type CDATA #REQUIRED>	10M

<!ATTLIST car make CDATA #FIXED "Ford">
<!ATTLIST car price CDATA #IMPLIED>
<car doors = "2" engine_type = "V8">
...
</car>

Declaring Entities :

Two kinds:

• A general entity can be referenced anywhere in the content of an XML document

Ex: Predefined entities are all general entities.

• A parameter entity can be referenced only in DTD.

• General Form of entity declaration.

<!ENTITY [%] entity_name "entity_value">

% when present it specifies declaration parameter entity

Example :

<!ENTITY jfk "John Fitzgerald Kennedy">

A reference to above declared entity: &jfk;

• If the entity value is longer than a line, define it in a separate file (an external text entity)

General Form of external entity declaration

<!ENTITY entity_name SYSTEM "file_location">

SYSTEM specifies that the definition of the entity is in a different file.

Example for parameter entity

<!ENTITY %pat "(USN, Name)">

<!ELEMENT student %pat; >

6. a) Declare an XML document containing data of 3 employees(Name, ID , Designation, Age, Phone ,address) and display this XML data using CSS with the following rules — 6M

ID- font size 28pts and color- red

Name and Designation - font size 18pts and color- blue

Age, phone, address- font size 15pts and color- black.

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet href="employee.css" type="text/css"?>
<!-- xml objects delclaration -->

<Employee>
  <Employee-info> Employee Information </student-info>

  <Employee1>
    <name>ABC</name>
    <ID>001</ID>
    <Designation>Manager</Designation>
    <Age>40</Age>
    <phone>225524525</phone>
    <address>Bangalore</address>
  </Employee1>

  <Employee2>
    <name>XXYZ</name>
    <ID>002</ID>
    <Designation>Design Engineer</Designation>
    <Age>45</Age>
    <phone>225524525</phone>
    <address>Bangalore</address>


  </Employee2>
```

```
   <Employee3>
      <name>LMN</name>
      <ID>003</ID>
      <Designation>Sales Manager</Designation>
      <Age>40</Age>
      <phone>225524525</phone>
      <address>Bangalore</address>


   </Employee3>
</Employee>

<!-- selecting the object style -->
Employee{display:block;}
Employee-info{display:block;color:red;text-align:center;}
Employee1{display:block;color:green;font-style:normal;}
Employee2{display:block;font-style:normal;}
Employee3{display:block;font-style:normal;}
ID{font-size:28px;color- red;}
name,Designation{font-size:18px;color- blue;}
Age,phone, address{font-size:15px;color- black;}
```

b) Apply XSLT styling to print the above data in tabular form (4M)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<head>
<style>
table,th,td
{
border:1px solid black;
border-collapse:collapse;
}
</style>
</head>
<body>
<h2>Employee Details</h2>
<table>
<tr bgcolor="#EEDD82" >
<th> Name </th>
<th> ID</th>
<th>Designation </th>
<th> Age </th>
<th> phone </th>
<th> Address </th>
</tr>
<xsl:for-each select="Employee/Employee-info">
<tr>
<td><xsl:value-of select="ID"/></td>
<td><xsl:value-of select="Name"/></td>
<td><xsl:value-of select="Designation"/></td>
<td><xsl:value-of select="Age"/></td>
<td><xsl:value-of select="phone
"/></td>
<td><xsl:value-of select="Address "/></td>
</tr>
</xsl:for-each>
```

```
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

7. Write a Perl program to read student details (name, usn, branch, semester) and store them into database and display the students who belong to ISE branch. Use HTML tags for displaying results — 10M

```
#!C:\xampp\perl\bin\perl
use CGI':standard';
use DBI;
print header();
print start_html(-title=>"Database",-bgcolor=>"white",-text=>"black");
#! Declare the driver,database, username and password
$db="anand";
$s="mysql";
$driver="DBI:$s:database=$db";
$user="root";
$pwd="";
    #! Connect to the database using driver, username and password
    $connection=DBI->connect($driver,$user,$pwd)or die $DBI::errstr;
        #! get parameter values form which has passed by html
        $name=param('name');
        $usn=param('usn');
                    $sem=param('sem');
                    $branch=param('branch');
          #! Write the query to insert the values to table
          $query="insert into student values('$name',$usn, $sem',$branch)";

              #Prepare and execute a query to insert the values
              $rs=$connection->prepare($query);
              $rs->execute();
              print "Data inserted successfully";
               #! Write a query to fetch the values form table
            $query="select * from student where branch=”ISE””;
              $rs=$connection->prepare($query);
              $rs->execute();
              #! Execute the query to fetch the values fromtable

          print"<h1> <center> List of the record in the database</center></h1>";

          print"<table border='2'> <tr><th>name<th>usn<th>sem<th>branch> </tr>";
          #! the records one by one until next reow is empty
      while(($name,$usn,$sem,$branch)=$rs->fetchrow())
      {
      print"<tr><td>$name</td><td>$usn</td>><td>$sem</td ><td>$branch</td ></tr>"
      }
            print "</table>";
          $rs->finish();
    $con->disconnect();
print end_html();
```