UNIT: B336 Advanced Internet Computing - Internal & External

TIME ALLOWED:- 3 hours plus 10 minutes reading time

INSTRUCTIONS:-

NAME:_____ STUDENT NUMBER:_____

SIGNATURE:_____

1. There are 21 pages in this exam manuscript, NOT including this cover page.
2. This exam consists of 15 questions, totalling 120 marks.
3. The marks for each question are indicated at the beginning of the question.
4. Enter the answers directly on this exam manuscript in the spaces provided under each question.

EXAMINATION AIDS

To be provided by the University

nil

To provided by the candidate

nil

SAMPLE EXAM ONLY

## Part I: Short and Medium-length Answers

1. What do the following acronyms stand for? (1 mark each, 5 marks total)

   a) XSLT

   b) SSI

   c) DTD

   d) WAE

   e) PCDATA

2. Where can you find the **official** reference sources for the following: (2.5 marks each, 10 marks total)

   a) Details of Perl LWP's `HTTP::Request` class

   b) Details of HTTP request message format

   c) Syntax for XML documents

   d) Complete set of WML tags

3. Explain the basic mechanism the Apache web server uses to restrict access to resources on its web-site. (5 marks)

4. Explain the following line from an Apache standard access_log file. You are expected to make reasonable educated guesses at what the fields are likely to be. (5 marks)

```
216.35.116.56 - - [20/Nov/2000:16:48:44 +0800]
  "GET /~jsmith/index.html HTTP/1.0" 200 9944
```

5. In the following two Perl example source code file, explain the sections which deals with an incoming HTTP request message which has a request line:
   `GET /units/b336/index.shtml HTTP/1.2`
   (10 marks)

Web.pm:

```
package Web;
# file: Web.pm
# Figure 15.1: Core Web Server Routines

# utility routines for a minimal web server.
# handle_connection() and docroot() are only exported functions

use strict;
use vars '@ISA','@EXPORT';
require Exporter;

@ISA = 'Exporter';
@EXPORT = qw(handle_connection docroot);

my $DOCUMENT_ROOT = '/home/www/htdocs';
my $CRLF = "\015\012";

sub handle_connection {
  my $c = shift;   # socket
  my ($fh,$type,$length,$url,$method);
  local $/ = "$CRLF$CRLF";   # set end-of-line character
  my $request = <$c>;        # read the request header

  return invalid_request($c)
    unless ($method,$url) =
                    $request =~ m!^(GET|HEAD) (/.*) HTTP/1\.[01]!;
  return not_found($c)
     unless ($fh,$type,$length) = lookup_file($url);
  return redirect($c,"$url/") if $type eq 'directory';

  # print the header
  print $c "HTTP/1.0 200 OK$CRLF";
  print $c "Content-length: $length$CRLF";
  print $c "Content-type: $type$CRLF";
  print $c $CRLF;

  return unless $method eq 'GET';

  # print the content
  my $buffer;
  while ( read($fh,$buffer,1024) ) {
    print $c $buffer;
  }
  close $fh;
}

sub lookup_file {
  my $url = shift;
  my $path = $DOCUMENT_ROOT . $url;             # turn into a path
  $path =~ s/\?.*$//;                           # get rid of query
  $path =~ s/\#.*$//;                           # get rid of fragment
  $path .= 'index.html' if $url=~m!/$!;         # get index.html
                                                # path ends in /
  return if $path =~ m!/\.\./!;                 # don't allow
```

```
                                                # relative paths (..)
  return (undef,'directory',undef) if -d $path;   # oops! a directory
  my $type = 'text/plain';                        # default MIME type
  $type = 'text/html'  if $path =~ /\.html?$/i;   # HTML file?
  $type = 'image/gif'  if $path =~ /\.gif$/i;     # GIF?
  $type = 'image/jpeg' if $path =~ /\.jpe?g$/i;   # JPEG?
  return unless my $length = (stat(_))[7];        # file size
  return unless my $fh = IO::File->new($path,"<");# try to open file
  return ($fh,$type,$length);
}

sub redirect {
  my ($c,$url) = @_;
  my $host = $c->sockhost;
  my $port = $c->sockport;
  my $moved_to = "http://$host:$port$url";
  print $c "HTTP/1.0 301 Moved permanently$CRLF";
  print $c "Location: $moved_to$CRLF";
  print $c "Content-type: text/html$CRLF$CRLF";
  print $c <<END;
<HTML>
<HEAD><TITLE>301 Moved</TITLE></HEAD>
<BODY><H1>Moved</H1>
<P>The requested document has moved
<A HREF="$moved_to">here</A>.</P>
</BODY>
</HTML>
END
}

sub invalid_request {
  my $c = shift;
  print $c "HTTP/1.0 400 Bad request$CRLF";
  print $c "Content-type: text/html$CRLF$CRLF";
  print $c <<END;
<HTML>
<HEAD><TITLE>400 Bad Request</TITLE></HEAD>
<BODY><H1>Bad Request</H1>
<P>Your browser sent a request that this server
does not support.</P>
</BODY>
</HTML>
END
}

sub not_found {
  my $c = shift;
  print $c "HTTP/1.0 404 Document not found$CRLF";
  print $c "Content-type: text/html$CRLF$CRLF";
  print $c <<END;
<HTML>
<HEAD><TITLE>404 Not Found</TITLE></HEAD>
<BODY><H1>Not Found</H1>
 <P>The requested document was not found on this server.</P>
</BODY>
</HTML>
END
}

sub docroot {
  $DOCUMENT_ROOT = shift if @_;
```

```
  return $DOCUMENT_ROOT;
}

1;
```

web_serial.pl:

```
#!/usr/bin/perl -w
# file: web_serial.pl
# Figure 14.2: The baseline server handles requests serially

use strict;
use IO::Socket;
use Web;

my $port = shift || 8080;
my $socket = IO::Socket::INET->new( LocalPort => $port,
             Listen    => SOMAXCONN,
             Reuse     => 1 )
                        or die "Can't create listen socket: $!";
while (my $c = $socket->accept) {
  handle_connection($c);
  close $c;
}
close $socket;
```

- BLANK PAGE –

6. What is the purpose of status codes sent from a web server to a web client? Give 2 example categories of HTTP status codes. (5 marks)

7. In the context of HTTP communications, what is a User Agent? (5 marks)

8. Find and describe the 10 mistakes in the following (supposedly) well-formed and valid XML document, which makes it violate the official W3C recommendations of XML version 1.0? Please **circle and number the appropriate sections of the code**, and **state what the mistakes are** either beside the numbered circles, or in the space after the code. (10 marks)

```
<?XML version="1.0" standalone="yes"?>
<!DTD DOCUMENT [
<!ELEMENT DOCUMENT (COURSE)*>
   <!ELEMENT COURSE (CODE, COURSENAME?, INFO*)>
      <!ELEMENT CODE (#PCDATA)>
      <!ELEMENT COURSENAME (#PCDATA)>
      <!ELEMENT INFO (LECTURER*,SEMESTER?,OPTIONAL?)>
         <!ELEMENT LECTURER (NAME, CONTACT*)>
            <!ELEMENT NAME (SURNAME, OTHERNAMES?)>
               <!ELEMENT SURNAME (#PCDATA)>
               <!ELEMENT OTHERNAMES (#PCDATA)>
            <!ELEMENT CONTACT (#PCDATA)>
         <!ELEMENT SEMESTER [ONE|TWO]>
         # Is the course optional?
         <!ELEMENT OPTIONAL EMPTY>
]>

<DOCUMENT>
   <COURSE>
      <CODE>B336</CODE>
      <COURSENAME>Advanced Internet Computing</COURSENAME>
      <INFO>
         <LECTURER>
            <NAME>
               <SURNAME>Hiew</SURNAME>
               <OTHERNAMES>Hong</OTHERNAMES>
               <OTHERNAMES>Liang</OTHERNAMES>
            </NAME>
            <CONTACT>h.hiew@murdoch.edu.au</CONTACT>
            <CONTACT>08 9360 6058</CONTACT>
         </LECTURER>
         <LECTURER>
            <NAME>Graham Mann</NAME>
         </LECTURER>
         <TEXTBOOK>None</TEXTBOOK>
         <OPTIONAL>Yes</OPTIONAL>
      </INFO>
   </COURSE>
</DOCUMENT>
<DOCUMENT>
   <COURSE>
      <CODE>B227</CODE>
      <INFO>
         <LECTURER>
            <NAME>
               <SURNAME>Cole</SURNAME>
               <OTHERNAMES>Peter</OTHERNAMES>
            </NAME>
            <CONTACT>p.cole@murdoch.edu.au</CONTACT>
         </LECTURER>
      </INFO>
   </COURSE>
</DOCUMENT>
</?XML>
```

- BLANK PAGE -

9. Write <u>one</u> XSLT style sheet which will convert of the following course1.xml to the course1.html, and course2.xml to course2.html. (5 marks)

course1.xml:

```
<?xml?>
<DOCUMENT>
  <COURSE>
    <CODE>B336</CODE>
    <LECTURE>Weds 12:30</LECTURE>
    <LECTURE>Fri 15:30</LECTURE>
  </COURSE>
</DOCUMENT>
```

course1.html

```
<HTML>
<BODY>
  <P>B336 lectures: <BR>
    <B>Weds 12:30</B> <BR>
    <B>Fri 15:30</B> <BR>
  </P>
</BODY>
</HTML>
```

course2.xml:

```
<?xml?>
<DOCUMENT>
  <COURSE>
    <CODE>B208</CODE>
    <LECTURE>Weds 17:30</LECTURE>
  </COURSE>
</DOCUMENT>
```

course2.html

```
<HTML>
<BODY>
  <P>B208 lectures: <BR>
    <B>Weds 17:30</B> <BR>
  </P>
</BODY>
</HTML>
```

10. What are the differences between parsing XML using DOM and parsing XML using SAX? (5 marks)

11. What mechanism resolves conflicts between tags with the same name used in different applications of XML? Use a simple XML code example to illustrate how it works. (5 marks)

12. How do WML and WMLScript fit in the WAP architecture? (5 marks)

13. What do the following WML and WMLScript do? (5 marks)

mystery.wml:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
  <card id="card1" title="Mystery" newcontext="true">
    <p>
    Input: <input format="*N" name="value" title="Value:"/><br/>
    Output: <u>$(final_value)</u>

    <do type="accept" label="Calculate">
      <go href="mystery.wmls#mystery('final_value',$(value))"/>
    </do>

    </p>
  </card>
</wml>
```

mystery.wmls:

```
extern function mystery(var1, var2) {

        var returnString ;

        if (var2 == 21) {
                 returnString = String.toString(var2);
        } else {
                returnString = "Big Deal!" ;
        }

        WMLBrowser.setVar(var1,returnString);
        WMLBrowser.refresh();
}
```

## Part II: Long Answers

14. Describe a simple HTTP request/response exchange between a web server and a web client, focusing on the format of the messages involved? (20 marks)

- BLANK PAGE –

15. What is involved in constructing a solution using XML? If necessary, use an example to illustrate your points. (20 marks)

- BLANK PAGE -

- END OF PAPER -