

Midterm  
**TI1506 Web and Database Technology**

Tuesday, 8 December 2015  
13.30-15.30

**INSTRUCTIONS:**

- This exam consists of 2 parts (DB and Web) and a total of 37 multiple-choice questions. All questions are worth an equal number of points.
- The usage of books, notes, old exams, and other written resources is explicitly **FORBIDDEN** during the exam. The use of electronic aids such as smart-phones, laptops, etcetera, is **ALSO NOT** allowed.
- There is only one right answer for each question. If you think there are more, pick the best one.
- You are not allowed to make corrections on the multiple-choice answer form (MAF). You are therefore advised to first mark the answers on this exam and later copy them to the MAF. If you need to make corrections anyway, ask for a new form and copy all your answers to it.
- You are not allowed to take the exam sheet with you after the exam. We will publish online the text of the exam together with its solutions.
- Note that the order of the answers on your MAF form is not always A-B-C-D.
- Be sure to fill in all header information on the MAF. Enter your *student number* on the form with digits as well as by filling the boxes.
- Sign the MAF. Without your signature, the form is not valid. Since you might forget this at the end, you are advised to do this at the start of the exam.

**Correct Answers Are Highlighted in Green**

## Part 1 – Web (23 questions)

Many of the following questions will be related to the **LOFI (Local Deals Finder)** Web application, which consists of a server-side and a client-side component. LOFI alerts you to interesting deals that shops and restaurants near you offer.

We employ state-of-the-art crowdsourcing to continuously fill our database with new deals from shops and restaurants. The client-side interface of our Web application is shown below. Clients (i.e. Web browsers) poll the server every 5 seconds for new deals. Clients then filter the deals to be shown to the user according to a number of conditions.



The three Web pages of our Web application are:

- overview: this page shows a list of all deals in the user's vicinity from shops and restaurants that are currently open; the list is automatically updated after each server poll (each deal has a certain expiration date after which the client no longer shows it); on the client, filtering is performed to only list those deals that are within a range of  $x$  meters from the user's current location and have a maximum price tag of  $y$ . Each user of LOFI can set  $x$  and  $y$  individually.
- details: when the user is interested in a particular deal, he clicks on the deal and is shown a page containing more details (location of the shop, a description of the item, opening hours of the shop, etc.). On this page, the user can also decide to "make the deal" and reserve the item.
- reserve: finally, if a user has indeed decided to reserve the item, he is redirected to the RESERVE page, where he fills in his details including name, address and an estimate of the number of minutes until picking up the item. The data is sent to the server, which forwards it to the respective shop or restaurant.

**+++ Note: Questions not concerning LOFI are marked as [General]. +++**

**QUESTION 1.** When inspecting the server-side source code of LOFI, you discover the code snippet below. What is the **main issue** of this piece of code?

```
1 var http = require('http'),
2   url = require('url');
3
4 var simpleHTTPResponder = function (req, res) {
5   var url_parts = url.parse(req.url, true);
6   if (url_parts.pathname === "/overview") {
7     res.writeHead(200, {
8       "Content-Type": "text/html",
9       "Content-Length": "50"
10    });
11    var content = retrieve_content();
12    //content now contains the generated OVERVIEW page
13    res.end(content);
14  }
15 };
16 var server = http.createServer(simpleHTTPResponder).listen(3000);
```

- A) The HTTP response is never sent.
- B) The HTTP response is missing the Status-Code header field.
- C) The HTTP response body will be truncated by the Web browser.
- D) The way the `simpleHTTPResponder` function is passed in as argument to `createServer` in line 16 is wrong (the server will not receive any HTTP requests).

**QUESTION 2.** What effects do Web caches have on the server hosting LOFI and on the clients using the LOFI application?

- A) The server's processing power will increase the larger the number of Web caches maintaining copies of LOFI. The clients will receive content faster.
- B) The server load will be significantly reduced the larger the number of Web caches maintaining copies of LOFI. The clients do not receive content any faster.
- C) The demand on the server will not change significantly. The clients do not receive content any faster.
- D) The server's distance delay and redundant data transfer will be reduced. The client's demand on the origin server will also be reduced.

**QUESTION 3.** Which of the following HTTP request method(s) is/are minimally required to access the full functionality of LOFI?

- A) GET
- B) GET, HEAD
- C) HEAD
- D) GET, POST

**QUESTION 4.** In the LOFI application, what is the best setting of the EXPIRES HTTP header field?

- A) EXPIRES is not a valid field in the HTTP header.
- B) In the HTTP response only, EXPIRES should be set to 5 seconds.
- C) In the HTTP request only, EXPIRES should be set to 5 seconds.
- D) In the HTTP request and response EXPIRES should be set to 5 seconds.

**QUESTION 5. [General]** Client and server can use HTTP request/response header fields to “negotiate” the form of the content. The server aims to send the content in a form the client understands. In order to achieve this, which header fields are client and server using?

- A) The server uses Accept-Encoding, the client uses Content-Encoding.
- B) The server uses Content-Encoding, the client uses Accept-Encoding.
- C) The server and the client both use Accept-Encoding.
- D) The server and the client both use Content-Encoding.

**QUESTION 6. [General]** How do Web caches employ the If-Modified-Since HTTP header to work efficiently?

- A) A Web cache sends an HTTP GET request to an origin server. In reply to this request, the origin server sends an HTTP response. If this response contains an If-Modified-Since header field with a value set to a date in the past, the Web cache does not store the response.
- B) A Web cache only sends an HTTP request to an origin server if the Web resource has been modified by the origin server in the meantime.
- C) A Web cache sends an HTTP GET request to an origin server with the If-Modified-Since header field set. In reply to this request, the origin server sends an HTTP response. This response only contains a content body if the resource has changed after the date found in If-Modified-Since.
- D) A Web cache only sends an HTTP request to an origin server when the Cache-Control and the If-Modified-Since dates coincide.

**QUESTION 7. [General]** Why is base-64 encoding employed in HTTP basic authentication?

- A) It ensures that the login and password cannot be decrypted easily by an attacker.
- B) It ensures that non-HTTP-compatible characters in the login and password are converted to HTTP compatible characters.
- C) It ensures that once the login and password information have been sent from the browser to the server, future HTTP requests to the same server do not require authentication.
- D) It ensures that the login and password fit into the available character limit of the WWW-Authenticate header field.

**QUESTION 8.** The LOFI application does not ask the user to login before using it. You are sharing your mobile phone with a friend. While you access LOFI with Mozilla Firefox, your friend uses Google Chrome. How is it possible for LOFI to reliably distinguish your usage of the application from your friend's usage on the same phone?

- A) The information in the HTTP request header field REFERER is sufficient.
- B) **Fat URLs can be used.**
- C) Client-side HTTP proxy tracking can be used.
- D) HTTPS can be used.

**QUESTION 9.** In LOFI, the user can change the maximum distance and the maximum price on the OVERVIEW page. Through which HTML5 functionality can we allow the user to change these numbers without resorting to HTML forms or JavaScript?

- A) **The attribute contenteditable.**
- B) The attribute formeditable.
- C) The attribute texteditable.
- D) It is not possible. Either HTML forms or JavaScript need to be used.

THIS QUESTION HAS BEEN CONSIDERED CORRECT FOR EVERY STUDENT

**QUESTION 10.** In the overview page of LOFI (and only in that page) you find this code snippet:

```
1 <!doctype html>
2 <html manifest="myman.appcache">
```

The file *myman.appcache* has the following content:

```
1 CACHE MANIFEST
2
```

How does this setup influence the capabilities of our LOFI application?

- A) When a user loads the page overview for the first time, the browser creates an appcache. Since no content is listed in *myman.appcache*, the appcache remains empty. It has no effect on the workings of LOFI.
- B) When a user loads the page overview for the first time, the browser creates an appcache. It caches the overview page but nothing else. When loading LOFI the next time, the overview page is served from the browser's appcache, while the details and reserve pages are requested from the server.
- C) When a user loads the page overview for the first time, the browser creates an appcache. It automatically discovers all resources that are part of the Web application and stores them in the appcache. When loading LOFI the next time, the browser serves all content from the appcache.
- D) **When a user loads the page overview for the first time, the browser creates an appcache. It caches the overview page but nothing else. When loading LOFI the next time, the browser only retrieves content from the appcache, it does not request missing content (for details and reserve) from the server.**

**QUESTION 11. [General]** After executing the JavaScript code snippet below, how many variables have a global scope?

```
1 var a = 5;
2 function outer(p){
3     function inner() {
4         a = p;
5         b = 12;
6     }
7     inner();
8     var c = 8;
9 }
10 outer(6);
11 a;b;c;p;
```

- A) 1
- B) 2
- C) 3
- D) 4

**QUESTION 12. [General]** After executing the JavaScript code snippet below what will be the output?

```
1 function Deal(m, i, e) {
2     this.message = m;
3     this.imgSrc = i;
4     this.endDate = e;
5     this.summarize = function() {
6         return this.message+" until "+this.endDate;
7     }
8 }
9 Deal.prototype.summarize = function() {
10     return this.message;
11 };
12 Deal.summarize = function() {
13     return this.message+"|"+this.endDate;
14 }
15 var d1 = new Deal("Toy kitchen","http://mytoys.nl/kitchen","12/12/2015");
16 var d2 = new Deal("Peppa Pig","http://mytoys.nl/peppa","23/12/2015");
17
18 d2.summarize = function() { return "["+this.message+"]"; }
19
20 d1.summarize();
21 d2.summarize();
```

- A)  
"Toy kitchen until 12/12/2015"  
"[Peppa Pig]"
- B)  
"Toy kitchen|12/12/2015"  
"Peppa Pig|23/12/2015"

- C) "Toy kitchen"  
"Peppa Pig"
- D) "Toy kitchen until 12/12/2015"  
"Peppa Pig|23/12/2015"

**QUESTION 13. [General]** After executing the JavaScript code snippet below, what will be the content of variable d?

```
1 function outer(X,Y) {  
2     var a = 1;  
3     var inner = function(b) {  
4         var c = X+Y;  
5         return c + a;  
6     }  
7     return inner;  
8 }  
9 var d = outer(2,3);
```

- A) undefined  
B) a function  
C) 6  
D) NaN

**QUESTION 14. [General]** After executing the JavaScript code snippet below what will be the console output?

```
1 var message = "Toy kitchen";  
2 var price = "89.90";  
3  
4 var deal1 = {  
5     message: "Peppa Pig",  
6     details: {  
7         price: "29.95",  
8         getPrice: function() {  
9             return this.price;  
10        }  
11    }  
12 }  
13  
14 var a = deal1.details.getPrice;  
15 a();
```

- A) a TypeError  
B) function() { return this.price; }  
C) "29.95"  
D) "89.90"

**QUESTION 15.** To start our application development, we built a simple HTML page, shown below. When you open this Web page in a JavaScript-enabled browser with Internet access,

what will be the list of shown deals?

```
1  <!doctype html>
2  <head>
3      <title>Local Deals Finder</title>
4      <script src="http://code.jquery.com/jquery-1.11.1.min.js">
5      </script>
6      <script>
7          $(document).ready(function () {
8              var el1 = document.getElementById("i22");
9              el1.innerHTML = "Toyota car";
10             });
11             var el2 = document.getElementById("i2");
12             el2.innerHTML = "Duftwasser";
13         </script>
14     </head>
15     <body>
16         <header>
17             <h1>lofi</h1><h2>Local Deals Finder</h2>
18         </header>
19         <div id="deals">
20             <ul>
21                 <li id="i1" data-id="122" data-latitude="12.43434"
22                     data-longitude="44.21" data-price="29.95">
23                     Toy car</li>
24                 <li id="i2" data-id="342" data-latitude="12.12332"
25                     data-longitude="44.09" data-price="9.99">
26                     Perfume</li>
27             </ul>
28         </div>
29         <div id="settings">
30             <!-- max. price and max. distance settings -->
31         </div>
32     </body>
33 </html>
```

- A)   
 Toy car  
 Perfume
- B)   
 Toyota car  
 Perfume
- C)   
 Toy car  
 Duftwasser
- D)   
 Toyota car  
 Duftwasser

**QUESTION 16.** In a prototype version of LOFI, we want to implement functionality in JavaScript that retrieves deals' details from the server when a user clicks on a deal. The code below contains a first implementation of this functionality. What is the **main issue** of this code?



```

1  <!doctype html>
2  <head>
3      <title>Local Deals Finder</title>
4      <script src="http://code.jquery.com/jquery-1.11.1.min.js">
5      </script>
6      <script>
7          var main = function() {
8              var list = document.getElementsByTagName("li");
9              for(var i=0; i<list.length; i++) {
10                 document.getElementsByTagName("li")[i].onclick = showDetails();
11             }
12         }
13         $(document).ready(main);
14         function showDetails() {
15             /* implementation of show details functionality */
16             console.log("Showing some details ...");
17         }
18     </script>
19 </head>
20 <body>
21     <header>
22         <h1>lofi</h1><h2>Local Deals Finder</h2>
23     </header>
24     <div id="deals">
25         <ul>
26             <li id="i1" data-id="122" data-latitude="12.43434"
27                 data-longitude="44.21" data-price="29.95">
28                 Toy car</li>
29             <li id="i2" data-id="342" data-latitude="12.12332"
30                 data-longitude="44.09" data-price="9.99">
31                 Perfume</li>
32         </ul>
33     </div>
34     <div id="settings">
35         <!-- max. price and max. distance settings -->
36     </div>
37 </body>
38 </html>

```

- A) Reloading the Web page  $n$  times will lead to  $n$  listeners being attached to the same list item.
- B) No event listeners will be attached to click events on list items.
- C) The JavaScript code will be executed before the DOM tree is loaded, the event listeners will be attached to the window object instead of to the list items.
- D) The method `document.getElementsByTagName()` does not exist leading to an error in the script.

**QUESTION 17.** On the server-side, the first node.js prototype of LOFI simply waits for HTTP requests and sends the contents of *deals.txt* (a plain text file with more than 50 deals, one per line) back to the clients. Or at least, that is the goal. The code does not work as intended. What is the issue with this code?

```

1 var http = require('http');
2 var fs = require('fs');
3 var server;
4 var filename = "deals.txt";
5 server = http.createServer(function (req, res) {
6     var deals;
7     //fs.readFile is an asynchronous function
8     fs.readFile(filename, function (err, fileContents) {
9         deals = fileContents;
10    });
11    res.writeHead(200, {
12        "Content-Type": "text/plain"
13    });
14    res.end(deals);
15
16 });
17 var port = 2333;
18 server.listen(port);

```

- A) The code is erroneous: `createServer` is not a method of the `http` module.
- B) The server starts but never sends back HTTP responses as reply to HTTP requests.
- C) The server starts but always sends back HTTP responses with an empty content body as reply to HTTP requests.
- D) Port 2333 is a so-called “system port” and can only be used by system processes. The server cannot be started on this port.

**QUESTION 18.** The first complete prototype of LOFI is now ready, with working server-side and client-side functionality. We relied on Ajax, to give the application a modern feel. Choose the correct order of the following steps in the interaction between client and server:

- (A) Render injected data
- (B) Execute client-side JavaScript
- (C) HTTP request for deals data
- (D) Send static files
- (E) HTTP request for overview

- A) E, B, D, C, A
- B) E, C, B, D, A
- C) E, B, D, C, A
- D) E, D, B, C, A

**QUESTION 19.** [General] Consider the following node.js script:

```

1 var express = require("express");
2 var url = require("url");
3 var http = require("http");
4 var app;
5
6 app = express();
7 http.createServer(app).listen(1033);
8
9 app.get("/", function(req, res) {
10     res.send("What?");
11 })
12
13 app.get("/greetings", function (req, res) {
14     var query = url.parse(req.url, true).query;
15     var name = ( query["name"]!=undefined) ? query["name"] : "Anonymous";
16     res.send("Greetings "+name);
17 });
18
19 app.get("/cheerio", function (req, res) {
20     res.send("Cheerio.");
21 });

```

It is started on the local machine and now in the browser you try to access the following URLs:

<http://localhost:1033/>  
<http://localhost:1033/GREETINGS>  
<http://localhost:1033/hello>  
<http://localhost:1033/cheerio?NAME=Delft>  
<http://localhost:1033/?name=Utrecht>  
<http://localhost:1033/greetings?NAME=Delft>

How many times do you receive a “Cannot GET /[name of route]” response?

- A) 0
- B) 1
- C) 2
- D) 3

**QUESTION 20. [General]** Through which mechanism can data be stored directly in CSS files?


- A) content attribute
- B) text attribute
- C) document attribute
- D) body attribute

**QUESTION 21. [General]** How does the browser render the following Web page?

```

1 <!doctype html>
2 <head>
3 <style>
4     main *:not(.deal) {
5         color:red;
6     }
7     main:not(#firstdeal) {
8         color:blue;
9     }
10 </style>
11 </head>
12 <body>
13 <main>
14 <h2>Deals</h2>
15 <p id="firstdeal">Toy kitchen</p>
16 <p class="deal">Perfume</p>
17 <p class="deal">Peppa Pig</p>
18 <p>Lawn mower</p>
19 </main>
20 </body>
21 </html>

```

Deals	Deals	Deals	Deals
Toy kitchen	Toy kitchen	Toy kitchen	Toy kitchen
Perfume	Perfume	Perfume	Perfume
Peppa Pig	Peppa Pig	Peppa Pig	Peppa Pig
Lawn mower	Lawn mower	Lawn mower	Lawn mower
A) 	B)	C)	D)

**QUESTION 22. [General]** Which of the following statements is true about CSS selectors?

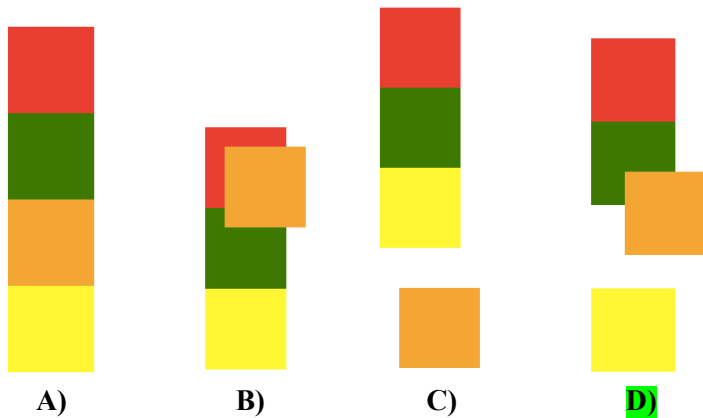
- A) The *tag* attribute can be added to any DOM element.
- B) A single DOM element cannot both contain a *class* and an *id* attribute.
- C) The *class* attribute can be added to any DOM element.
- D) A single DOM element cannot contain multiple classes.

**QUESTION 23. [General]** How is the following HTML page rendered?

```

1 <!doctype html>
2 <head>
3   <style>
4     div {
5       height:50px;
6       width:50px;
7     }
8     #b1 { background-color:red;}
9     #b2 { background-color:green;}
10    #b3 {
11      background-color:orange;
12      position:relative;
13      bottom:20px;
14      left:20px;
15    }
16    #b4 { background-color:yellow;}
17  </style>
18 </head>
19 <body>
20   <div id="b1"></div>
21   <div id="b2"></div>
22   <div id="b3"></div>
23   <div id="b4"></div>
24 </body>
25 </html>

```



## Part 2 – Database

**QUESTION 24.** Which of the following statements about DBMS systems **is correct**?

- A) The overhead for providing concurrency control in DBMS system does not compensate for the additional cost of DBAs.
- B) DBMS systems must provide specialized search techniques to speed up data storage on disk.
- C) The lifetime of the managed data is always longer than the lifetime of the application using the DBMS system.
- D) A DBMS system must provide facilities for recovering from hardware or software failures.

**QUESTION 25.** Which of the following statements about conceptual data models is **wrong**?

- A) Conceptual data models provide details of how concepts are stored in the computer storage media.
- B) Conceptual data models provide the means to achieve data abstraction.
- C) Conceptual data models provide concepts that are close to the way many users perceive data.
- D) Conceptual data models use concepts such as entities, attributes, and relationships.

**QUESTION 26.** Which of the following statements about *relations* in the relational data model **are true**?

- 1. Tuples in a relation are, by default, ordered according to their order of insertion.
  - 2. An n-tuple is an ordered list of  $n$  values.
  - 3. Each value in an n-tuple is *atomic*; that is, it is divisible into sub-atomic components.
  - 4. NULL values represent the values of attributes that might be unknown.
- A) [1] and [3]
  - B) [2] and [3]
  - C) [2] and [4]
  - D) [1] and [4]

**QUESTION 27.** Which of the following **are** elements of an SQL schema?

- 1. Tables
  - 2. Authorization Grants
  - 3. Catalog
  - 4. Custom Domains
- A) Only [1] and [4]
  - B) Only [2] and [3]
  - C) [1], [2], and [4]
  - D) Only [3] and [4]

**QUESTION 28.** What is the Cartesian product  $D_1 \times D_2$  of the following domains?

D<sub>1</sub> = ( 'WebDB', 'OOP' )  
D<sub>2</sub> = ( 'John', 'Maria', 'Anna' )

- A) {< 'WebDB', 'OOP' >, < 'John', 'Maria'>, < 'John', 'Anna'>, < 'Maria', 'Anna'>}
- B) {< 'WedDB', 'John'>, < 'OOP', 'Maria'>, {< 'WedDB', 'Maria'>, < 'OOP', 'Anna'> } }
- C) {< 'John', 'WedDB'>, < 'Maria', 'WedDB'>, < 'Anna', 'WedDB'>, < 'John', 'OOP'>, < 'Maria', 'OOP'>, < 'Anna', 'OOP'>}
- D) {< 'WedDB', 'John'>, < 'WedDB', 'Anna'>, < 'OOP', 'Maria'>, < 'OOP', 'John'>, < 'WedDB', 'Maria'>, < 'OOP', 'Anna'>}

The following questions are related to the **Chinook** database schema, reported in appendix A.

SUGGESTION: detach pages 19 to 22 of this document, so that you can easily have them in front of you at all times.

The database can be used to manage a digital media store. It includes tables for artists, albums, media tracks, genres, playlists, invoices and customers.

Before answering the following questions: analyse the database schema, explore the properties of the tables, and understand how the tables relate to each other. Identify primary and foreign keys, and understand their constraints.

**QUESTION 29.** Which of the following statements about the miniworld modeled by the **Chinook** database is correct?

- A) A *Track* can be bought multiple times within the same invoice
- B) A *Track* can belong to multiple *Albums*
- C) Two *Artists* can be authors of the same *Album*
- D) A *Customer* can be provided support by multiple *Employees*

**QUESTION 30.** Which of the following constraints can be enforced by the **Chinook** database?

- A) Two *Artists* cannot publish an *Album* with the same title
- B) A *Track* cannot be included multiple times in the same *Playlist*.
- C) *Tracks* in the same album must have the same *MediaType*.
- D) *Tracks* must be always sold at the same price

**QUESTION 31.** Assume the **Chinook** database has been enriched with the following constraints

```
ALTER TABLE `Track` MODIFY `GenreId` INT NOT NULL;

ALTER TABLE `Track` ADD CONSTRAINT `FK_TrackGenreId`
  FOREIGN KEY (`GenreId`) REFERENCES `Genre` (`GenreId`)
  ON DELETE NO ACTION ON UPDATE NO ACTION;
```

Among the following queries, which one does not always return the same result set as other ones?

1. SELECT \*

- ```
FROM Genre JOIN Track ON Genre.GenreId = Track.GenreId
```
2. `SELECT *`  
`FROM Genre LEFT JOIN Track ON Genre.GenreId = Track.GenreId`
  3. `SELECT *`  
`FROM Genre RIGHT JOIN Track ON Genre.GenreId = Track.GenreId`
- A) [1]  
 B) [2]  
 C) **[3]**  
 D) They all return the same result set

**QUESTION 32.** Assume the **Chinook** database has been enriched with the same constraints as defined in the previous question. Which of the following statements **is true**?

- A) Deleting a tuple from the *Genre* table will cause the corresponding attribute in the *Track* table to be deleted.  
 B) Deleting a tuple from the *Genre* table will cause the corresponding tuples in the *Track* table to be deleted.  
 C) **Deleting a tuple from the *Genre* table will cause the database to reject the operation.**  
 D) Deleting a tuple from the *Genre* table will cause no action from the database.

**QUESTION 33.** Which of the following SQL queries **is guaranteed** to always produce a result-set containing **ALL** the tuples from the *Track* table of the **Chinook** database?

- A) `SELECT DISTINCT AlbumId, Name FROM Track`  
 B) **`SELECT DISTINCT TrackId, Name FROM Track`**  
 C) `SELECT AlbumId FROM Track`  
 D) **`SELECT * FROM Track WHERE Name IS NOT NULL`**

BOTH ANSWERS WERE COUNTED AS CORRECT

The following questions are related to the **Chinook** database state, reported in Appendix B. The database is an instantiation of the **Chinook** database schema in Appendix A.

When answering the following questions: analyse the database state, and relate the values contained in it with the properties of the database schema.

**QUESTION 34.** Which answer contains the correct output of the following SQL query?

```
SELECT T1.TrackId, T2.TrackId
FROM Track AS T1 INNER JOIN Genre
      ON T1.GenreId = Genre.GenreId
      LEFT JOIN Track AS T2 ON T2.GenreId = Genre.GenreId
WHERE T1.TrackId <> T2.TrackId AND T1.AlbumId = T2.AlbumId
      AND T1.AlbumId < 3
```



- A) {<1,3>, <4,6>, <6,4>, <3,1>}
- B) {<1,3>, <4,6>}
- C) {<1,3>, <2,4>, <2,6>}
- D) {<1,5>, <2,4>, <4,2>, <5,1>, <2,6>, <6,2>}

**QUESTION 35.** Which answer reports the correct output of the following SQL query?

```
SELECT TrackId, COUNT(PlaylistId)
FROM PlaylistTrack
GROUP BY TrackId
HAVING PlaylistId = 2
```

- A) {NULL}
- B) {<2, 34>}
- C) {<2,8>}
- D) The database returns an error because the query is not correct

**QUESTION 36.** Which answer reports the correct output of the following SQL query?

```
SELECT E1.EmployeeId, E2.EmployeeId
FROM Employee AS E1 INNER JOIN Employee AS E2
      ON E1.EmployeeId = E2.ReportsTo
WHERE E2.ReportsTo IS NULL
```

- A) { }
- B) {NULL}
- C) {<1,2>}
- D) {<2,1>}

**QUESTION 37.** Which of the following statements best describe the result set produced by following SQL query?

```
SELECT ar1.name, ar2.name
FROM Album AS al1, Album AS al2, Artist AS ar1, Artist AS ar2
WHERE al1.title LIKE "A%" and al2.title LIKE "A%"
      AND al1.AlbumId = al2.AlbumId
      AND al1.ArtistId = ar1.ArtistId
      AND al2.ArtistId = ar2.ArtistId
```

- A) The names of all the pairs of *Artists* that published an *Album* having a title that contains the letter A.
- B) The permutations of the names of all *Artists* that published an *Album* having a title that contains the letter A.
- C) The names of all *Artists*, returned 2 at a time, that published an *Album* having a title that starts with the letter A.
- D) The permutations of all *Artists* that published an *Album* having a title that starts with the letter A.

THIS QUESTION HAS BEEN CONSIDERED CORRECT FOR EVERY STUDENT



## Appendix A – Chinook Database Schema

```
CREATE TABLE `Album`
(
    `AlbumId` INT NOT NULL,
    `Title` NVARCHAR(160) NOT NULL,
    `ArtistId` INT NOT NULL,
    CONSTRAINT `PK_Album` PRIMARY KEY (`AlbumId`)
);

CREATE TABLE `Artist`
(
    `ArtistId` INT NOT NULL,
    `Name` NVARCHAR(120),
    CONSTRAINT `PK_Artist` PRIMARY KEY (`ArtistId`)
);

CREATE TABLE `Customer`
(
    `CustomerId` INT NOT NULL,
    `FirstName` NVARCHAR(40) NOT NULL,
    `LastName` NVARCHAR(20) NOT NULL,
    `Company` NVARCHAR(80),
    `Address` NVARCHAR(70),
    `City` NVARCHAR(40),
    `State` NVARCHAR(40),
    `Country` NVARCHAR(40),
    `PostalCode` NVARCHAR(10),
    `Phone` NVARCHAR(24),
    `Fax` NVARCHAR(24),
    `Email` NVARCHAR(60) NOT NULL,
    `SupportRepId` INT,
    CONSTRAINT `PK_Customer` PRIMARY KEY (`CustomerId`)
);

CREATE TABLE `Employee`
(
    `EmployeeId` INT NOT NULL,
    `LastName` NVARCHAR(20) NOT NULL,
    `FirstName` NVARCHAR(20) NOT NULL,
    `Title` NVARCHAR(30),
    `ReportsTo` INT,
    `BirthDate` DATETIME,
    `HireDate` DATETIME,
    `Address` NVARCHAR(70),
    `City` NVARCHAR(40),
    `State` NVARCHAR(40),
    `Country` NVARCHAR(40),
    `PostalCode` NVARCHAR(10),
    `Phone` NVARCHAR(24),
    `Fax` NVARCHAR(24),
    `Email` NVARCHAR(60),
    CONSTRAINT `PK_Employee` PRIMARY KEY (`EmployeeId`)
);

CREATE TABLE `Genre`
(
    `GenreId` INT NOT NULL,
    `Name` NVARCHAR(120),
    CONSTRAINT `PK_Genre` PRIMARY KEY (`GenreId`)
);

CREATE TABLE `Invoice`
```

```

(
    `InvoiceId` INT NOT NULL,
    `CustomerId` INT NOT NULL,
    `InvoiceDate` DATETIME NOT NULL,
    `BillingAddress` NVARCHAR(70),
    `BillingCity` NVARCHAR(40),
    `BillingState` NVARCHAR(40),
    `BillingCountry` NVARCHAR(40),
    `BillingPostalCode` NVARCHAR(10),
    `Total` NUMERIC(10,2) NOT NULL,
    CONSTRAINT `PK_Invoice` PRIMARY KEY (`InvoiceId`)
);

CREATE TABLE `InvoiceLine`
(
    `InvoiceLineId` INT NOT NULL,
    `InvoiceId` INT NOT NULL,
    `TrackId` INT NOT NULL,
    `UnitPrice` NUMERIC(10,2) NOT NULL,
    `Quantity` INT NOT NULL,
    CONSTRAINT `PK_InvoiceLine` PRIMARY KEY (`InvoiceLineId`)
);

CREATE TABLE `MediaType`
(
    `MediaTypeId` INT NOT NULL,
    `Name` NVARCHAR(120),
    CONSTRAINT `PK_MediaType` PRIMARY KEY (`MediaTypeId`)
);

CREATE TABLE `Playlist`
(
    `PlaylistId` INT NOT NULL,
    `Name` NVARCHAR(120),
    CONSTRAINT `PK_Playlist` PRIMARY KEY (`PlaylistId`)
);

CREATE TABLE `PlaylistTrack`
(
    `PlaylistId` INT NOT NULL,
    `TrackId` INT NOT NULL,
    CONSTRAINT `PK_PlaylistTrack` PRIMARY KEY (`PlaylistId`, `TrackId`)
);

CREATE TABLE `Track`
(
    `TrackId` INT NOT NULL,
    `Name` NVARCHAR(200) NOT NULL,
    `AlbumId` INT,
    `MediaTypeId` INT NOT NULL,
    `GenreId` INT,
    `Composer` NVARCHAR(220),
    `Milliseconds` INT NOT NULL,
    `Bytes` INT,
    `UnitPrice` NUMERIC(10,2) NOT NULL,
    CONSTRAINT `PK_Track` PRIMARY KEY (`TrackId`)
);

```

## Appendix B – Chinook Database Instance

*With respect to Appendix A, the schema of some table has been simplified to improve readability.*

### Album

| AlbumId | Name                     | ArtistId |
|---------|--------------------------|----------|
| 1       | Thriller                 | 1        |
| 2       | Bad                      | 1        |
| 3       | Appetite for destruction | 2        |
| 4       | Use your illusion        | 2        |

### Artist

| ArtistId | Name            |
|----------|-----------------|
| 1        | Micheal Jackson |
| 2        | Guns n Roses    |

### Track

| TrackId | Name              | AlbumId | MediaTypeId | GenreId | Composer        | Milliseconds | Bytes    | UnitPrice |
|---------|-------------------|---------|-------------|---------|-----------------|--------------|----------|-----------|
| 1       | Beat it           | 1       | 1           | 1       | Quincy Jones    | 258000       | 5160000  | 1.99      |
| 2       | Billie Jean       | 1       | 1           | 3       | Quincy Jones    | 294000       | 5920000  | 1.99      |
| 3       | Thriller          | 1       | 1           | 1       | Quincy Jones    | 357000       | 7140000  | 1.99      |
| 4       | Bad               | 2       | 1           | 3       | Micheal Jackson | 247000       | 7410000  | 2.49      |
| 5       | Man in the mirror | 2       | 1           | 1       | Micheal Jackson | 320000       | 9600000  | 2.49      |
| 6       | Smooth Criminal   | 2       | 1           | 3       | Micheal Jackson | 259000       | 7770000  | 2.49      |
| 7       | Paradise City     | 3       | 5           | 2       | Axl Rose        | 406000       | 20300000 | 2.49      |
| 8       | Rocket Queen      | 3       | 5           | 2       | Axl Rose        | 377000       | 18850000 | 2.49      |
| 9       | November Rain     | 4       | 5           | 2       | Axl Rose        | 537000       | 26850000 | 2.49      |
| 10      | Coma              | 4       | 5           | 2       | Axl Rose        | 613000       | 30650000 | 2.49      |

### Genre

| GenreId | Name |
|---------|------|
| 1       | Pop  |

|   |      |
|---|------|
| 2 | Rock |
| 3 | Misc |

### MediaType

| MediaTypeId | Name                        |
|-------------|-----------------------------|
| 1           | MPEG audio file             |
| 2           | Protected AAC audio file    |
| 3           | Protected MPEG-4 video file |
| 4           | Purchased AAC audio file    |
| 5           | AAC audio file              |

### PlaylistTrack

| PlaylistId | TrackId |
|------------|---------|
| 1          | 1       |
| 3          | 1       |
| 1          | 2       |
| 3          | 2       |
| 1          | 3       |
| 3          | 3       |
| 1          | 4       |
| 3          | 4       |
| 1          | 5       |
| 3          | 5       |
| 1          | 6       |
| 3          | 6       |
| 2          | 7       |
| 3          | 7       |
| 2          | 8       |
| 3          | 8       |
| 2          | 9       |
| 3          | 9       |
| 2          | 10      |
| 3          | 10      |

### Playlist

| PlaylistId | Name         |
|------------|--------------|
| 1          | my pop music |

|   |            |
|---|------------|
| 2 | let's rock |
| 3 | all music  |

### InvoiceLine

| InvoiceLineId | InvoiceId | TrackId | UnitPrice | Quantity |
|---------------|-----------|---------|-----------|----------|
| 1             | 1         | 1       | 1.99      | 1        |
| 2             | 1         | 2       | 1.99      | 1        |
| 3             | 1         | 3       | 1.99      | 1        |
| 4             | 1         | 4       | 1.99      | 1        |
| 5             | 1         | 5       | 1.99      | 1        |
| 6             | 1         | 6       | 1.99      | 1        |
| 7             | 2         | 7       | 2.49      | 1        |
| 8             | 2         | 8       | 2.49      | 1        |
| 9             | 3         | 9       | 2.49      | 1        |
| 10            | 3         | 10      | 2.49      | 1        |

### Invoice

| InvoiceId | CustomerId | InvoiceDate         | BillingAddress         | BillingState | BillingCountry | BillingPostalCode | Total |
|-----------|------------|---------------------|------------------------|--------------|----------------|-------------------|-------|
| 1         | 1          | 2011-11-03 00:00:00 | 1.8154 2nd Street West | PA           | US             | 19446             | 11.94 |
| 2         | 1          | 2011-12-10 00:00:00 | 1.8154 2nd Street West | PA           | US             | 19446             | 4.98  |
| 3         | 1          | 2011-12-11 00:00:00 | 1.8154 2nd Street West | PA           | US             | 19446             | 4.98  |

### Customer

| CustomerId | FirstName | LastName | Company | Address                | City      | State | Country | Email         | SupportRedId |
|------------|-----------|----------|---------|------------------------|-----------|-------|---------|---------------|--------------|
| 1          | Jason     | Anderson | IBM     | 1.8154 2nd Street West | Landsdale | PA    | US      | jan@yahoo.com | 2            |

### Employee

| EmployeeId | LastName | FirstName | Position | ReportsTo | BirthDate           | HireDate            | Address                | City              | Email              |
|------------|----------|-----------|----------|-----------|---------------------|---------------------|------------------------|-------------------|--------------------|
| 1          | Boss     | M. R.     | CEO      | NULL      | 1961-11-03 00:00:00 | 2001-01-01 00:00:00 | 4.4431 Hillcrest Drive | Fort Walton Beach | mrboss@company.com |

|   |     |      |                      |   |                     |                     |                 |       |                  |
|---|-----|------|----------------------|---|---------------------|---------------------|-----------------|-------|------------------|
| 2 | Doe | John | Customer Service guy | 1 | 1982-06-08 00:00:00 | 2010-11-03 00:00:00 | 5.6091 Broadway | Tampa | jdoe@company.com |
|---|-----|------|----------------------|---|---------------------|---------------------|-----------------|-------|------------------|