UNIVERSITY OF TORONTO SCARBOROUGH

WINTER 2016 EXAMINATIONS

CSC B20H3
Duration — 3 hours

No Aids Allowed

Student Number: |_|_|_|_|_|_|_|_|_|_|_|_|_|

Last Name: _____

First Name: _____

---

*Do **not** turn this page until you have received the signal to start.*
(In the meantime, please fill out the identification section above, and read the instructions below *carefully*.)

This final examination consists of 3 questions on 15 pages (including this one). When you receive the signal to start, please make sure that your copy of the examination is complete. Answer each question directly on the examination paper, in the space provided. (If you need more space for one of your solutions, use the blank page and indicate **clearly** which part of your work should be marked.)

Be aware that concise, well thought-out answers will be rewarded over long rambling ones. Also, unreadable answers will be given zero (0) so write legibly.

In the programming questions, assume all input is valid. Make sure to indent your code properly. Your code will be marked based on correctness and style.

1: _____/ 30

2: _____/ 25

3: _____/ 45

TOTAL: _____/100

Good Luck!

Question 1. [30 MARKS]

Consider the following set of relations for a music database. The database stores an artist (or band) `id` and `name`, `album id` and `title`, a relation to tie artists to albums as well as a relation to list the tracks associated with each album. Answer each subquestion. You may create VIEWS if that simplifies your answer. If there is no possible solution, write "NOT POSSIBLE".

Database relations:

artist(id, name)

album(id, title)

artist_album(artist_id, album_id)

track(track_title, album_id, num)

Part (a) [2 MARKS]

Underline the primary keys.

Part (b) [2 MARKS]

Find all the album `id`s for albums having the `title` "*Greatest Hits*".

Soln. `select id from album where title = 'Greatest Hits';`

Part (c) [3 MARKS]

Find the artist `id` of the artists who have an album entitled "*Greatest Hits*".

Soln. `select artist_id from album join artist_album on id = album_id where title = 'Greatest Hits';`

Part (d) [3 MARKS]

Find the names of the artist's that have an album title containing the word "Unplugged".

Soln. `select name from artist join album join artist_album on artist_id = artist_album.artist and album.id = artist_album.album_id and album.id = album_id where album.title like '%Unplugged%';`

Part (e) [4 MARKS]

Return for each artist the average number of tracks on their albums.

Soln. use group by

Question 1. (CONTINUED)

Database relations:

artist(id, name)

album(id, title)

artist_album(artist_id, album_id)

track(track_title, album_id, num)

Part (f) [4 MARKS]

Return all the tracks from the '*Greatest Hits*' album by the artist *Queen* sorted by increasing track number. Create a `VIEW` called CDs containing artist `name`, album `title` and `album_id` to simplify your solution.

Soln.
```
create view CDs as (select name, title, album_id from artist join album join
artist_album on artist_id = artist_album.artist_id and album.id = artist_album.album_id
)

select track_title from CDs join track on CDs.album_id = track.album_id where album.title
= 'Greatest Hits' and artist.name = 'Queen' order by track.num;
```

Part (g) [4 MARKS]

Find all artist names who have produced more than 5 albums. You may use the `VIEW CDs` even if you were unable to write it. You may assume no two artists have the same name.

Soln.
```
select names from CDs group by names having count(album_id) >= 5;
```

Part (h) [4 MARKS]

Find those album titles with more than 15 tracks on them. Do not use the `HAVING` clause.

Soln.
```
select title from ((select album_id, count(album_id) as C from tracks) as
T ) join album on album.id = T.album_id where C>15;
```

Part (i) [4 MARKS]

Sometimes different albums by the same artist have tracks in common (same title). Find all `artist_ids` and their `track_titles` that belong to more than one album.

Question 2. [25 MARKS]

Short answer questions.

Part (a) [3 MARKS]

Explain the three different levels of data abstraction of a database.

Soln. Physical Level
o Lowest level, how the data are actually stored.
o Usually in complex low-level data structures.
Logical Level
o What data are stored in the database and what relationships exist between the data.
o Implementing the simple structure of the logical level may require complex physical low level structures.
o Users of the logical level don?t need to know about this.
o We refer to this as the physical data independence.
View Level:
o Highest level of abstraction - describes only a small portion of the database
o Allows user to simplify their interaction with the database system.
o Can have many views.

Part (b) [4 MARKS]

List two SQL operations which do not exist in MySQL. Choose one of them and explain how you would implement the operation in MySQL.

Soln. INTERSECT, DIFFERENCE. We can for intersect we can...

Part (c) [4 MARKS]

List two advantages, and two disadvantages of client-side scripting.

Soln.

Advantages include: better responsiveness (faster response time), offline applications, offloads network traffic and server load

Disadvantages include: scripts are readable thus harder to protect proprietary code (IP), security risks posed by code from untrusted sources, page download time increased by scripts

Other answers, if plausible, may also be acceptable

Part (d) [2 MARKS]

List two advantages associated with use of the jQuery library, compared with use of plain JavaScript.

Soln. Possible answers - need 2:

abstracts (hides) browser differences; expressiveness makes code smaller, simpler and faster to develop; simple/powerful selector system for identifying document/DOM objects to be acted upon; chaining of actions improves code clarity and efficiency; simple event handling mechanism including Ajax callback handlers. Other answers possible ?

Part (e) [6 MARKS]

Write a comment for each of the following PHP code snippits:

```
foreach ($y as $z) {
      $s = explode(",", $z);
      $m = $s[0];
      $n = $s[1];
 }
/*




*/

$w = file('x.txt');
/*




*/

$p = glob($q . "/r*.txt");
/*



*/

$p is assigned a list of the file names in directory $q that match pattern
r*.txt (the name ?r? followed by any sequence of characters and then ?.txt?  */
```

```
read contents of file 'x.txt' into array $w with one line in x.txt per array entry
 in $w
 */

 /* iterate over array $y, with individual elements referenced as $z, splitting
$z into an array using comma field separators, then assign the 1st element
of array $s to $m and the 2nd element to $n. */
```

Part (f) [3 MARKS]

List three benefits associated with use of an external CSS stylesheet as compared with CSS styles defined within an HTML document.

Soln. - An external stylesheet is **reusable** across multiple documents (DRY)
- Easier to **maintain** as change to a single CSS file potentially affects many referencing HTML documents
- More **efficient**, since only have to download the external CSS file once if it is referenced by multiple HTML documents vs downloading each copy of internally-defined CSS

Part (g) [3 MARKS]

Describe the effect of this CSS declaration (which element(s) does it affect and how?):

```
<style type="text/css">
  #happy  p  .day  { color: blue }
</style>
```

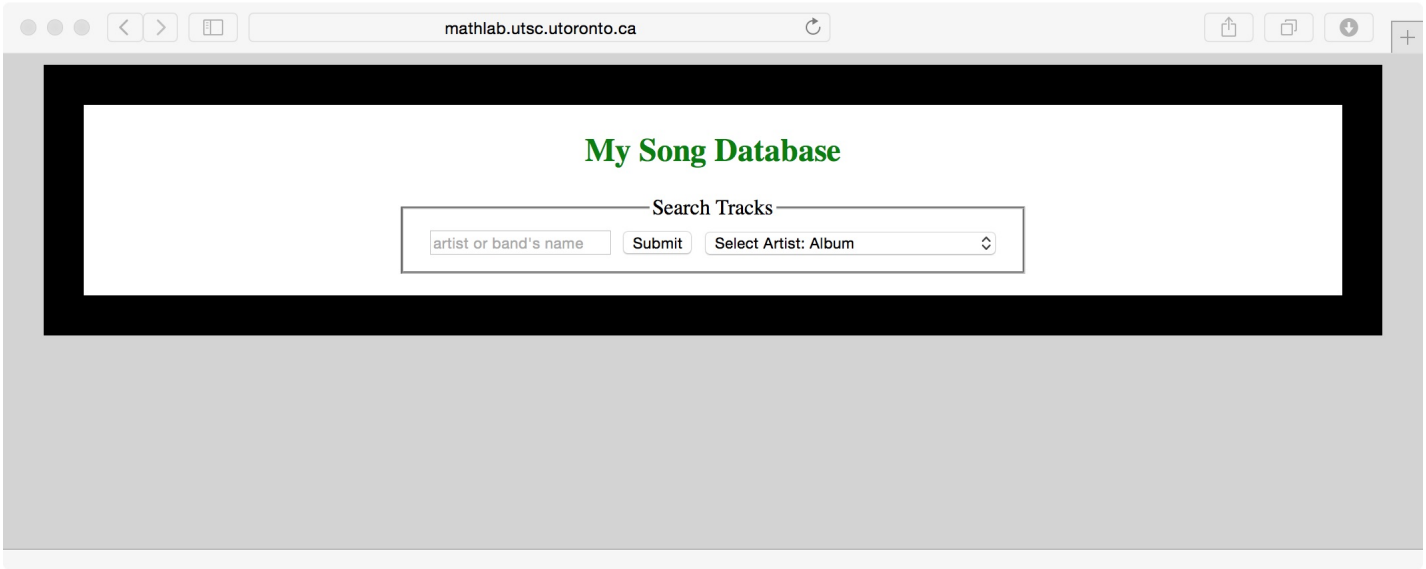
Soln. Apply blue color to elements with class ?day?, nested within (descendants of) ?p? elements, nested within (descendants of) an element with id=?happy?.

Question 3. [45 MARKS]

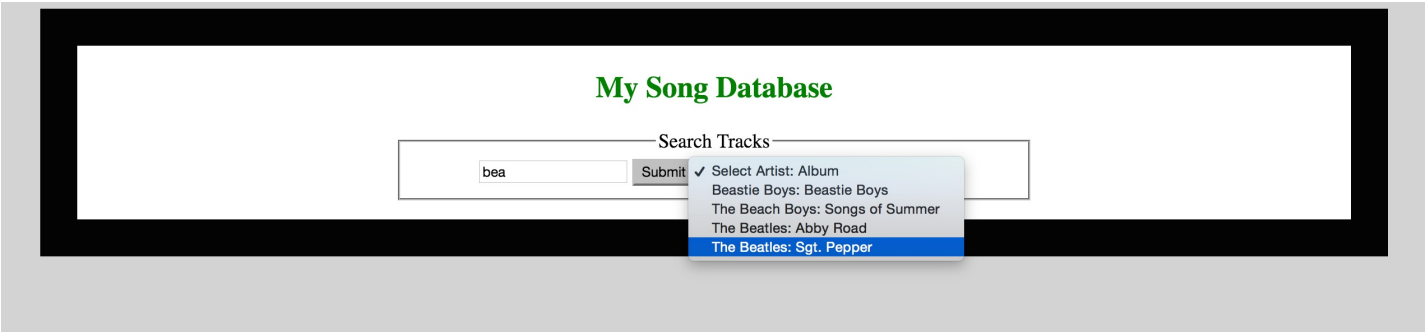
This question has multiple parts. You may complete later parts even if you could not complete the first parts.

Assume you have a music database as described in the SQL portion of the exam. We will create a web application similar to that of A3 that asks the user for a musical artist/band name or portion of a name. The user will then be provided with a drop down menu to select an album from an artist matching the supplied name. When an album is selected, a request is made to a webpage with PHP content which loads a page listing the tracks of the requested album. Here are some sample screen shots of the web application:

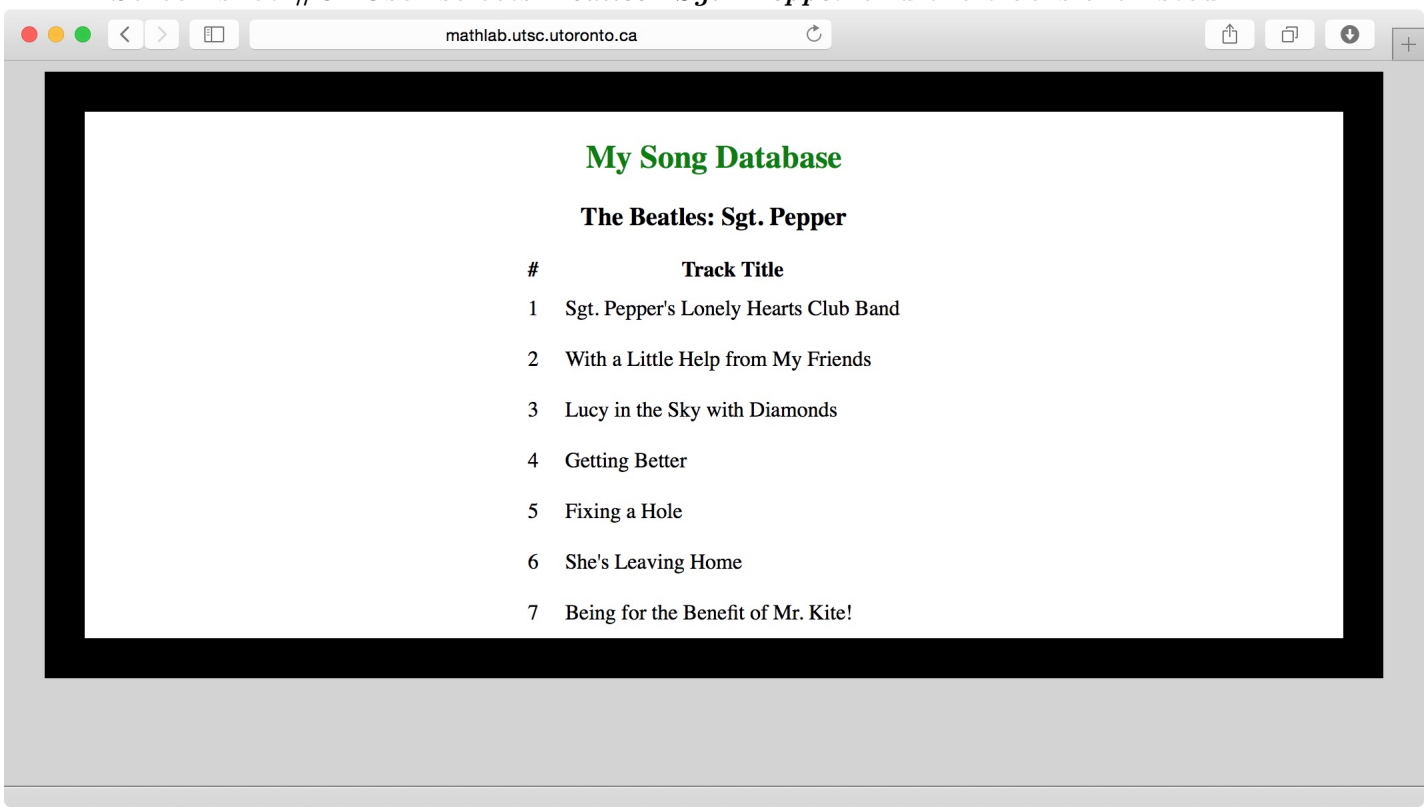
Screen shot #1: User enters artist/band name or partial name:



Screen shot #2: User enters *bea* and the matching bands are listed in alphabetical order with albums in the drop down menu:



Screen shot #3: User selects *Beatles: Sgt. Pepper* and the tracks are listed:



Part (a) [10 MARKS]

Like A3, a web service called `artists.php` is invoked to fill in the drop down menu (screenshot #2). The html code for the main page (screenshot #1) is listed here for your convenience:
Fill in the missing code for `artists.php` where indicated:

```
<?php
    /*include 'config.php';  // DB settings including credentials


        // connect to the database
        $dbh = mysqli_connect($dbhost, $dbuser, $dbpass, $dbname);

        // output an error message if connection fails
    if (!$dbh) {
            echo "DB connect error";
            die("Connection failed.");
    }

        // get the artists name entered by the user
        $artist = $_REQUEST["artist"];


        // Define an sql query to find all albums by a particular artist
        $sql = "SELECT artist, title, album_id FROM artist join album join artist_album
        on album_id = album.id and artist_id = artist.id WHERE name LIKE
        '%$artist%' ORDER BY artist, album;";
```

```
        // execute the sql statement
        $stmt = mysqli_query($dbh, $sql);

        $runs = array();
        // create an array containing all the results
        while ($row = mysqli_fetch_assoc($stmt)) {
                $runs[] = $row;
        }

        mysqli_close($conn);

        // encode query result as a JSON string and return it
        $json = json_encode($runs);
        print $json;
```

Part (b) [10 MARKS]

The file `songs.js` takes the JSON query results from `artists.php` and inserts them into the drop down selection menu. The format of the JSON data will look like this:

```
[ {"album":"Beastie Boys","artist":"Beastie Boys","id":"4"},
 {"album":"Songs of Summer","artist":"The Beach Boys","id":"3"},
 {"album":"Abby Road","artist":"The Beatles","id":"1"},
 {"album":"Sgt. Pepper","artist":"The Beatles","id":"2"}]
```

Fill in the missing code as indicated by the `<=== ADD CODE HERE` labels.

```
function searchAlbums() {
 /* Retrieve the user-selected option value (album_id) and text
    (artist/band name: album).  Store these values in hidden fields of
    artist_form.  Submit artist_form, after turning off our handler
    of submit events, so that we don't cycle back to the handler.
 */
 var album_id = $(this).val();
 var band_album = $("#artist option:selected").text();
 $('#album_id').val(album_id);  // set hidden album_id field
 $('#band_album').val(band_album);  // set hidden artist_name field
 $('#artist_form').off('submit');  // turn off submit handler to avoid recapture
 $('#artist_form').submit();  // submit artist_form
};

function selectAlbum(event) {
 /* Intercept artist_form submit event, and issue Ajax request to artists.php
    for list of artists who match the user-entered (possibly incomplete)
    name.  Display resulting JSON array as a sequence of
    HTML option values with value taken from the JSON id field and text
    content taken from the catenated JSON "artist name: album".
 */
 event.preventDefault();  // prevent default submit behaviour for form
 $.getJSON("artists.php",
```

```
   { 'artist':  $('#artist').val()},  // pass artist_form-field values
    function(data) {

    /* prepare content of HTML select input with HTML id=album
    (see main page code), assume artists.php does
    not return an error. Iterate through JSON result array, appending
    an HTML option for each with text "artist: album" and value "id".
     */

     $.each(data, function(index,item) {
       $('#album').append(
          $('<option class="search_result"></option>')
             .text(item.artist + ":  " + item.album).val(item.id)
       );
     //});  // end of $.each()
    })
};

$(document).ready(function() {
 // bind event handlers for artist_form submit, and select-menu change
 $('#artist_form').submit(selectAlbum);  // handler for artist_form submit

 $('#album').live('change', searchAlbums);  // handler for select change event
});
```

Part (c) [13 MARKS]

The final track results as shown in the third screen shot are displayed by calling `search.html` with parameters `album_id` and `band_album`.

Fill in the missing PHP for `search.html` as indicated:

```
<!DOCTYPE html>
<html>
 <head>
  <title>My Song Database</title>

  <!-- Links to jquery, js, css files etc  -->
        ...omitted...
 </head>
 <body>
  <div id="frame">
<?php

    include 'config.php';  // DB settings including credentials

    $conn = mysqli_connect($dbhost, $dbuser, $dbpass, $dbname);
    if (!$conn) {
```

```
        die("Connection failed: " . mysqli_connect_error());
    }

// get album_id parameter and band_album parameter values
// band_album contains "artist name: album name"
$id = $_REQUEST["album_id"];
$band_album = $_REQUEST["band_album"];

    // split the band_album into band/artist name and album title
    $artist = explode(";", $band_album)[0];
    $title = explode(";", $band_album)[1];

    // Use the album idea to select the track titles from the album
    // sorted by increasing track number
$query = "SELECT track_title FROM tracks WHERE
        album_id = $id ORDER BY num INC;"


$results = mysqli_query($conn, $query);
if (!$results){
        echo("Error Description: " . mysqli_error($conn));
} else {
        if (mysqli_num_rows($results) > 0) {
// non-empty result case follows

?>
<head>
  <title>My Song Database</title>
</head>

<body>
  <h2>My Song Database</h2>

           <!-- put artist: Album in h3 header  -->
  <h3><?= $band_album ?></h3>
  <table>
    <tr><th>#</th><th>Track</th></tr>
      <?php
          $i = 1;  // counter to label results in table
          while ($row = mysqli_fetch_assoc($results)) {
      ?>
      <tr>
          <td> <?= $i ?> </td>
          <td> <?= $row["track_title"] ?> </td>  <!-- track title -->
      </tr>

      <?php
              $i++;  // increment counter for next table row
```

```
              }  // end while
        ?>
      </table>
      <?php
      }  // end if results
    }   // end else no error

    mysqli_close($conn);
    ?>
</body>

  </div> <!-- end of #frame div -->
 </body>
</html>
\end{verbatim}
\else
\fi
\newpage

\subquestion{12} % CSS
\normalsize
In this question, you will write the CSS to format the output of the web application - the lis
resembling that displayed in the third screen shot.  Additional information is given in commen

\ifsolution
\begin{Verbatim}[frame=single]
/* page body is different color than content panel (#frame) */
body { background-color: lightgray;
           text-align: center;
         }

/* content frame takes up 90% of the window, left and right margins should
auto adjust when window resized, background colour is white, and the
thickness of the black border is 30px */
#frame {
        width: 90%;
        margin-left:auto;
        margin-right:auto;
        background-color: white;
        border: 30px solid black;
}


h2 {
        color: green;
}

/*  */
```

```
table {
        margin-left:auto;
        margin-right:auto;
        text-align: left;
}
td {
        padding: 10px;
}
th {
        text-align: center;
        font-weight: bold
}
```

Total Marks = 100