1520 Recitation 12
Review Exam

1. Answer the following questions using **map**, **filter** and **reduce** and the dataset below:

```
mario_kart =    [
                  {
                    "name": "Bowser",
                    "size": 10,
                    "origin": "Bowser Kingdom",
                    "times": [124, 182, 96, 25]
                  },
                  {
                    "name": "Waluigi",
                    "size": 8,
                    "origin": "Mushroom Kingdom",
                    "times": [148, 201, 107, 32]
                  },
                  {
                    "name": "Toad",
                    "size": 1,
                    "origin": "Mushroom Kingdom",
                    "times": [105, 170, 98, 33, 67]
                  },
                  {
                    "name": "Mii",
                    "size": 3,
                    "origin": "System",
                    "times": [111, 199, 88, 31]
                  }
                ]
```

1.a) Create an array of unique origins (in JS and Python):

JS:

```
var unique_origins = Array.from(new Set(mario_kart.map(x => x.origin)));
```

Python:

```
unique_origins = list( set(map(lambda x: x[“origin”], mario_kart) )
```

1.b) Create an array of average times per player (in JS and Python)

JS:

```
var avg_times = mario_kart.map(x => x.times.reduce((time, total)=> total + time) /
                                                    x.times.length );
```

Python:

1.c) Create an array of players bigger than 6 (in JS and Python)

JS:

```
var big_people = mario_kart.filter(x => x.size > 6);
```

Python:

```
big_people = filter(lambda x: x[“size”] > 6, mario_kart)
```

1.d) Create an array of prefix sums (successive summations) similar to the following output:

```
prefix_sums = [10, 18, 19, 22]
```

JS:

Python:

2. Consider the following AJAX code:

```
var xhttp = new (0);
xhttp.onreadystatechange = function() {
    if (this.readyState == (1) && this.status == (2)) {
       // Action to be performed when the document is read;
       var data = (3)
    }
};
xhttp.open("GET", "filename", true);
xhttp.send();
```

2.a) Fill in the blanks with reasons:

(0): XMLHttpRequest()
Reason:

(1) 4
Reason:

(2) 200
Reason:

(3) (hint: the returned data is JSON): JSON.parse(this.responseText);

Reason:

2.b) Let us assume you want to send the following key-value pairs using the AJAX code above:

```
"first_name"="Bla", "second_name"="Kla"
```

How would you change the above example code to add these key-value pairs in your AJAX request? (You only need to write the code line which you will modify)

2.b.i) If request method is of type GET:

```
xhttp.open("GET", "filename?first_name='Bla'&second_name='Kla'",
        true);
```

2.b.ii) If request method is of type POST:

```
xhttp.send("first_name='Bla'&second_name='Kla'");
```

2.c) Consider:

```
xhttp.open("GET", "filename", true);
```

2.c.i) What does the 3rd parameter used for? What does true or false mean for this parameter?

True means the request is asynchronous (non-blocking), and false means the request is synchronous (blocking).

2.c.ii) Should we normally use it as true or false? Why?

We should use it as true. Because, if blocking, then our app will not do anything else until we get a response from the server. Our app may become unresponsive and slow.

3. Which of the following HTTP method(s) should be idempotent in nature?

   A - GET

   B - DELETE

   C - POST

   D - HEAD

4. Which of the following HTTP Status code means CREATED, when a resource is successful created using POST or PUT request?

   A - 200

   B - 201

   C - 204

   D - 304

5. Which of the following HTTP Status code means INTERNAL SERVER ERROR, states that server has thrown some exception while executing the method?

   A - 500

B - 401

C - 404

D - 409

6. Which of the following is true about REST:

   A – Both Server and Client should be able to understand and utilize the representation format of the resource.

   B – Format should be able to represent a resource completely. For example, a resource can contain another resource. Format should be able to represent simple as well as complex structures of resources.

   C – A resource can have a linkage to another resource, a format should be able to handles such situations.

   D – All of the above.

   E – None of the above

7. What URI stands for?

   A - Unit Resource Identifier

   B - Uniform Representation Identifier

   C - Uniform Resource Identifier

   D - None of the above.

8. Your team is building an application and you are reviewing the functional specifications. Your team must include a stockticker in the UI that displays the companys stock price every 15 minutes, and include the capability to do partial saves of base objects as users step through a data entry wizard. You want to use the same approach for both requirements to make it easier to add functionality and maintain it going forward. What approach should you use?

   A - Use JavaScript to refresh the page every 15 minutes and to manage whether wizard buttons are enabled or disabled.

B - Use AJAX to make asynchronous calls to the server on a timer for the stock price and to automatically save the base objects as the user navigates through the wizard.

C - Use jQuery to refresh the page every 15 minutes and to manage whether wizard buttons are enabled or disabled.

D - Use data validation annotations on the model to ensure that the stock price is validated every 15 minutes and that the client saves the base object information after every wizard step.

E - C and A both

F - All of the above

9. You are creating a web application that will be accessed by a large number of traditional consumers. If you need to be able to access state information on the client side in JavaScript, where can you store it? (Choose all that apply.)

   A - LocalStorage

   B - QueryString

   C - ViewState

   D - Cookies

   E - B and D both

   F - All of the above

10. Dwight from DunderMifflin just came up with a new app to expand the company's business portfolio by opening a new e-commerce platform, DunderMifflin Online Paper Shop. Jim, as usual, pranked Dwight by deleting part of his code base. Dwight is furious and cannot think properly. He needs your help to fix the app.

    Relationships in the app:

    i) A **user** can have one or many **addresses**
    ii) An **address** can belong to only one **user**
    iii) A **user** can have zero or many **items** in their **cart**
    iv) An **item** can be in zero or many **people's carts**

Part of the code (flask models) is below:

```
db = SQLAlchemy()

cart = db.Table('cart',
        db.Column([1a], db.Integer, db.ForeignKey([1b]), primary_key=[1c]),
        db.Column([2a], db.Integer, db.ForeignKey([2b]), primary_key=[2c])
)

class User(db.Model):
        id = db.Column([3a], primary_key=[3b])
        username = db.Column([4a], nullable=[4b])
        email = [5a]
        pw_hash = [5b]
        cart = db.relationship([6a])
        [X]

        def __init__(self, username, email, pw_hash):
                self.username = username
                self.email = email
                self.pw_hash = pw_hash

        def __repr__(self):
                return '<User {}>'.format(self.username)

class Address(db.Model):
        id = db.Column([7a])
        street = db.Column([7b])
        city = db.Column([7c])
        state = db.Column([7d])
        zip_code = db.Column([7e])
        [X]

        def __init__(self, street, city, state, zip_code):
                self.street = street
                self.city = city
                self.state = state
                self.zip_code = zip_code

class Item(db.Model):
        id = db.Column([8a])
        name = db.Column([8b])
        in_stock = db.Column([8c])
```

```
        def __init__(self, name, in_stock):
                self.name = name
                self.in_stock = in_stock
```

10 Answers:

```
db = SQLAlchemy()

cart = db.Table('cart',
        db.Column('user_id', db.Integer, db.ForeignKey('user.id'), primary_key=True),
        db.Column('item_id', db.Integer, db.ForeignKey('item.id'), primary_key=True)
)

class User(db.Model):
        id = db.Column(db.Integer, primary_key=True)
        username = db.Column(db.String(24), nullable=False)
        email = db.Column(db.String(80), nullable=False)
        pw_hash = db.Column(db.String(128), nullable=False)
        cart = db.relationship('Item', secondary=cart, lazy='subquery',
backref=db.backref('users', lazy=True))

        def __init__(self, username, email, pw_hash):
                self.username = username
                self.email = email
                self.pw_hash = pw_hash

        def __repr__(self):
                return '<User {}>'.format(self.username)


class Address(db.Model):
        id = db.Column(db.Integer, primary_key=True)
        street = db.Column(db.String(50), nullable=False)
        city = db.Column(db.String(50), nullable=False)
        state = db.Column(db.String(50), nullable=False)
        zip_code = db.Column(db.String(50), nullable=False)
        user_id = db.Column(db.Integer, db.ForeignKey('user.id'))

        def __init__(self, street, city, state, zip_code):
                self.street = street
                self.city = city
                self.state = state
                self.zip_code = zip_code

class Item(db.Model):
```

```
id = db.Column(db.Integer, primary_key=True)
name = db.Column(db.String(50), nullable=False)
in_stock = db.Column(db.Boolean, nullable=False)

def __init__(self, name, in_stock):
        self.name = name
        self.in_stock = in_stock
```