# Advance Web App Midterm 201x

## Question 1(a)

In developing a Laravel app of a shopping mall directory, four models - **Tenant, Category, Floor, and Lot**. A tenant belongs to exactly one category and may occupy one or more floors and each floor contains many tenants. A tenant may occupy more than one lot while each lot can only be occupied by one tenant. Define all the relationship methods for 4 of the model classes.

```
class Tenant extends Model{
    public function category(){
        return $this->belongsTo(App\Category);
    }

    public function floor(){
        return $this->belongsToMany(App\Floor);
    }

    public function lot(){
        return $this->hasMany(App\Lot);
    }
}

class Category extends Model{
    public function tenant(){
        return $this->hasMany(App\Tenant);
    }
}

class Floor extends Model{
    public function tenant(){
        return $this->belongsToMany(App\Tenant);
    }
}

class Lot extends Model{
    public function tenant(){
        return $this->belongsTo(App\Tenant);
    }
}
```

## Question 1(b)

Define the validation rules to validate input for the **Tenant** model of the abovementioned app in (a) based on the requirements specified in the table below.

| Attribute | Rules |
|---|---|
| tenant_code | Required, unique, minimum 2 and maximum 3 characters |
| name | Required, maximum length of 20 characters |
| contact_person | Required, maximum length of 30 characters |
| phone | Required, matches the regular expression: ^([0-9]{2,3})-([0-9]{6,8})$ |
| email | Required, must be a valid email address |

```
$request->validate([
    'tenant_code' => 'required|unique|min:2|max:3',
    'name' => 'required|max:20',
    'contact_person' => 'required|max:30',
    'phone' => 'required|regex: ^([0-9]{2,3})\-([0-9]{6,8})$',
    'email' => 'required|email'
])
```

# Question 2(a)

Construct code segment to illustrate the logic how you would check if a user is authenticated.

```
// To login,

if(Auth::attempt(['email'=>$email, 'password'=>$password])){
    return view('users.index');
}

// To check whether the user is authenticated
if(Auth::check()){
    // User is logged in
}
```

# Question 2(b)

Construct code segment to illustrate how you would protect a route using the auth middleware.

```
Route::resource('/home', 'HomeController')->middleware('auth');
```

# Question 2(c)

Construct code segment to illustrate how you protect an entire controller using the **auth:api** middleware

```
// Within the controller

public function __construct(){
    $this->middleware('auth:api');
}
```

# Question 2(d)

Define the following gates to authorize user actions.

(i) Check if the user is allowed to create articles

```
Gate::define('user-create', function($user){
    // Assume the user have a field 'canCreateArticles' and it is a boolean
    return $user->canCreateArticles;
})
```

(ii) Check if the user is allowed to edit and delete own articles.

```
Gate::define('user-edit', function($user, $article){
    return $user->id == $article->user_id;
})

Gate::define('user-delete', function($user, $article){
    return $user->id == $article->user_id;
})
```

(iii) Check if the user is allowed to edit and delete all articles

```
Gate::define('user-delete-all', function($user){
    // Assume the user have a field 'isAdmin' and it is a boolean
    return $user->isAdmin;
})
```

---