Angular
Lab 4: Reactive Forms
Exercise Solutions

1 Lab setup

2 Generating a new Angular project

3 Adding a basic form control using FormControl

Q1. Create a radio button with a random value and bind it to a FormControl instance property in the component

Q2: Use interpolation to display the value of the radio button in the template itself.

app.component.ts

```
// Q1
language = new FormControl('');
```

app.component.html

```
<!-- Q1 -->
<input type="radio" id="html" [formControl]="language" value="HTML">
<label for="html">HTML</label><br>

<!-- Q2 -->

<p>The value of the radio button is : {{ language.value }}</p>
```

4 Using FormGroup to group related form controls

Q1. In the template, add in a new numeric input field for age just after the email field in the main FormGroup. Bind this element to a new FormControl instance in the component.

Q2. In the template, add in a new text input field for zipcode just after the state field in the nested FormGroup. Bind this element to a new FormControl instance in the component.

Q3: Use interpolation to display the values of these two new fields in the template itself.

app.component.ts

```
  profileForm = new FormGroup({
    firstname: new FormControl(''),
    lastname: new FormControl(''),
    email: new FormControl(''),
    // Q1
    age: new FormControl(0),
    address: new FormGroup({
      street: new FormControl(''),
      city: new FormControl(''),
      state: new FormControl(''),
      // Q2
      zipcode: new FormControl('')
    })
  });
```

app.component.html

```
  <!-- Q1 -->
  <p>
    <label for="age">Age :</label>
    <input type="number" id="age" formControlName="age">
  </p>

    <!-- Q2 -->
    <p>
      <label for="zipcode">Zip Code</label>
      <input type="text" id="zipcode" formControlName="zipcode">
    </p>



<!-- Q3 -->

<p>The age is : {{ profileForm.value.age }}</p>

<p>The zipcode in the address is :  {{ profileForm.value.address.zipcode
}}</p>
```

5 Submitting the form and accessing form and form control values

Q1: Repeat Q1 and Q2 from the previous section. Add in statements in the component method that handles the submitted form data in order to obtain the value for the age and zipcode form elements.

app.component.ts

```
    // Q1
    console.log('The age is ', this.profileForm.value.age);

    console.log('The zipcode is ', this.profileForm.value.address.zipcode);
```

6 Updating all or a portion of a FormGroup

Q1: Repeat Q1 and Q2 from the Section 4. Add in additional logic to the component method that uses setValue in order to provide new values for the zipcode and age controls.

Q2: Add in additional logic to the component method that uses patchValue in order to provide new values for the zipcode and age controls.

app.component.ts

```
profileForm = new FormGroup({
  firstname: new FormControl(''),
  lastname: new FormControl(''),
  email: new FormControl(''),
  // Q1
  age: new FormControl(0),
  address: new FormGroup({
    street: new FormControl(''),
    city: new FormControl(''),
    state: new FormControl(''),
    // Q1
    zipcode: new FormControl('')
  })
});


// Q1
setContentForEntireForm() {
  // Create an object with properties
  // that match the structure of the form exactly

  let supermanProfile = {
    firstname : 'Clark',
    lastname: 'Kent',
    email: 'superman@gmail.com',
    age: 44,
    address : {
      street : '99 Louis Lane',
      city : 'Metropolis',
      state : 'New York',
      zipcode: '9999',
    }

  };
```

```
    this.profileForm.setValue(supermanProfile);

  }


  // Q2
  setContentForPartOfForm() {

    let updatedProfile = {
      firstname : 'Tony',
      lastname: 'Stark',
      age : 56,
      address : {
        street : '100 Stark Industries',
        zipcode : '4444',
      }

    };

    this.profileForm.patchValue(updatedProfile);

  }
```

app.component.html

```
  <!-- Q1 -->
  <p>
    <label for="age">Age :</label>
    <input type="number" id="age" formControlName="age">
  </p>

    <!-- Q1 -->
    <p>
      <label for="zipcode">Zip Code</label>
      <input type="text" id="zipcode" formControlName="zipcode">
    </p>
```

7 Control status for a FormControl and FormGroup

No exercises here

8 Validating form controls

Q1: Add an additional validator to the first name field so that it only contains letters (no numbers, spaces or any other characters). Hint: Angular - Validators

app.component.ts

```
    // Q1
    firstname: new
FormControl('',[Validators.required,Validators.minLength(3),
Validators.pattern('[a-zA-Z]*')]),
```

For more info on regular expressions in JavaScript and how to use them:
https://www.javascripttutorial.net/javascript-regular-expression/
https://www.w3schools.com/jsref/jsref_obj_regexp.asp

9 Display validation error messages

Q1: Add three validators to the job field:

- Make it required
- Make it have a maximum of 10 characters
- Make it only contain letters (no numbers, spaces or any other characters). Hint: Angular - Validators

Q2: Add appropriate conditional logic to display validation error messages for the job field.

app.component.ts

```
    // Q1
    job: new
FormControl('',[Validators.required,Validators.maxLength(10),Validators.patt
ern('[a-zA-Z]*')])
```

app.component.html

```
  <!-- Q2: Validation error message for job -->
  <div *ngIf='job?.invalid && (job?.dirty || job?.touched)'
style="color:red">
    <div *ngIf="job?.errors?.['required']">
      Job is required.
    </div>
    <div *ngIf="job?.errors?.['maxlength']">
      Job cannot be more than 10 characters long.
    </div>
    <div *ngIf="job?.errors?.['pattern']">
      Job can only contain alphabets
    </div>
  </div>
```

10 Using FormBuilder as a shortcut for creating FormGroup

No exercises here

11 Using FormArray to build dynamic forms

Q1: Add another FormArray called `addresses` to the profile FormGroup items of the main profiles FormArray in the component. This FormArray `addresses` will contain FormGroup items as elements, where each FormGroup item has the following FormControls:

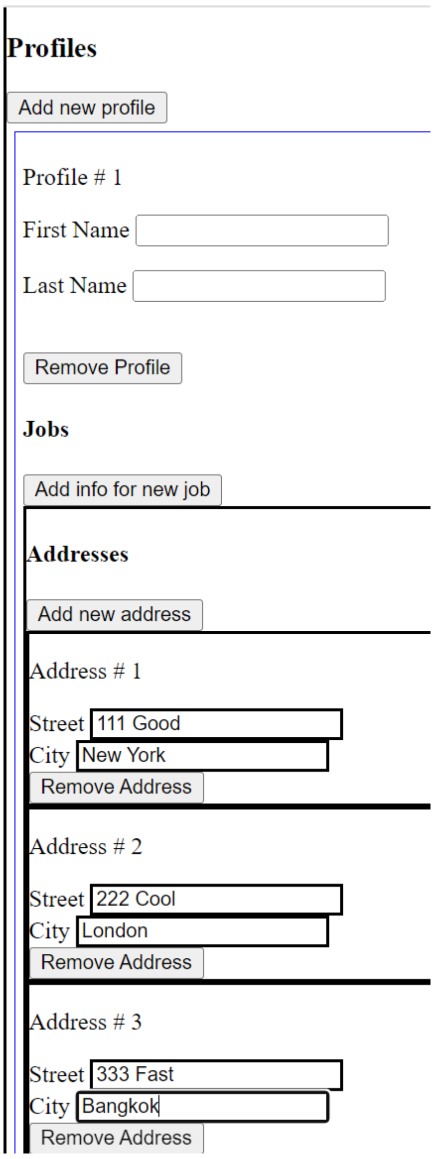
- Street: string
- City: string

Provide the required methods to

- include the addresses FormArray into the main profile FormGroup
- access the addresses FormArray
- create a new FormGroup item to place into the FormArray
- add this new FormGroup item into the FormArray
- remove an existing FormGroup item from the FormArray

Q2: Add the required statements in the template to support the items created in Q1 that allow the user to add and remove new addresses from a given top-level profile

Sample output from adding 3 new addresses to a single profile



Profiles

Add new profile

Profile # 1

First Name

Last Name

Remove Profile

Jobs

Add info for new job

Addresses

Add new address

Address # 1

Street 111 Good
City New York
Remove Address

Address # 2

Street 222 Cool
City London
Remove Address

Address # 3

Street 333 Fast
City Bangkok
Remove Address

app.component.ts

```
// Q1
addresses(profileIndex: number) {
  return this.profiles.at(profileIndex).get('addresses') as FormArray;
}

// Q1
createNewProfile(): FormGroup {
  return this.fb.group({
    firstname: [''],
    lastname: [''],
    jobs: this.fb.array([]),
    addresses: this.fb.array([])
  });
}

// Q1
createNewAddress(): FormGroup {
  return this.fb.group({
    street: [''],
    city: [''],
  });
}

// Q1
addAddress(profileIndex: number) {
  this.addresses(profileIndex).push(this.createNewAddress());
}

// Q1
removeAddress(profileIndex: number, addressIndex: number) {
  this.addresses(profileIndex).removeAt(addressIndex);
}
```

app.component.html

```
        <!-- Q2 -->
        <div formArrayName="addresses">
          <h4>Addresses</h4>
          <button type="button" (click)="addAddress(profileIndex)">Add new
address</button>

          <div *ngFor="let address of addresses(profileIndex).controls; let
addressIndex = index">

            <div [formGroupName]="addressIndex">
              <p>Address # {{addressIndex + 1}}</p>

              <label for="street">Street </label>
              <input type="text" id="street" formControlName="street" />
              <br>
              <label for="city">City </label>
              <input type="text" id="city" formControlName="city" />
              <br>
              <button (click)="removeAddress(profileIndex, addressIndex)">
                Remove Address
              </button>
              <br>
          </div>

        </div>
      </div>
```

12 Adding other form control elements (radio buttons, checkboxes, etc) to a reactive form

No exercises