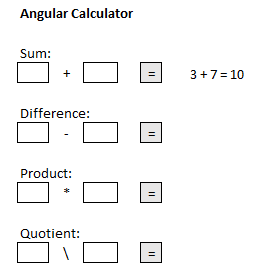
Due date to be announced.

* Please do not include your node\_modules folder. One mark will be deducted if it is included.
* Worth 10% of course mark.

It is okay to repeat your code for this assignment – actually I highly recommend just getting the project done rather than making a graceful solution. It is better to learn the basics at this point.

For this project, using Angular you will create a calculator on one page that looks like the following:



The calculator has these features:

* Clicking on the equals sign button clears all text inputs and outputs and displays the result for the current operation only. (5 marks)
* Each input implements validation to ensure the value is numeric (ie. 3 or 3.5 is valid but ‘ABC’ is not valid). If a text input is invalid an error message is displayed for the specific control set and the button is disabled. The button is only enabled when the text field is numeric. (4 marks)



* Text boxes are aligned and sized in a professional manner. Decimal places in the results are rounded to 2 digits. (1 mark)
* You do not need to validate empty fields.

**Tip 1:**

Instead of using the *template* option in the components annotation to write your HTML and directives, you can include these in a separate html file which is referenced using the *templateUrl* option. For example, if you had a file named **/app/app.html** you could reference it with this line of code:

templateUrl: './app.html'

**Tip 2:**

The HTML input type really needs to be an input of **'text'** type.  HTML does have several other input types and while they are helpful they don't let you handle validation with Angular consistently at the application level. Once the 'text' type is used for the input element, the AppComponent class can be setup like the following:

export class AppComponent {

public result:number;     // \*\*\* If we use 'this' we must have a class level declaration.

public Operand1:string;   // \*\*\* Same as above.

public Operand2:string;   // \*\*\* Same as above.

  doAddition() {

    this.result=Number(this.Operand1) + Number(this.Operand2);

  }

}​​

**Tip 3:**

It will be much easier and feasible at this stage of learning if you create four separate blocks of code in your HTML – one for each operation. Creating four separate blocks to handle the math operations is encouraged.