Challenge du weekend

Perfect Numbers
Determine if a number is perfect, abundant, or deficient based on Nicomachus' (60 - 120 CE) classification scheme for natural numbers.

The Greek mathematician Nicomachus devised a classification scheme for natural numbers, identifying each as belonging uniquely to the categories of perfect, abundant, or deficient based on their aliquot sum. The aliquot sum is defined as the sum of the factors of a number not including the number itself. For example, the aliquot sum of 15 is (1 + 3 + 5) = 9

Perfect: aliquot sum = number
6 is a perfect number because (1 + 2 + 3) = 6
28 is a perfect number because (1 + 2 + 4 + 7 + 14) = 28
Abundant: aliquot sum > number
12 is an abundant number because (1 + 2 + 3 + 4 + 6) = 16
24 is an abundant number because (1 + 2 + 3 + 4 + 6 + 8 + 12) = 36
Deficient: aliquot sum < number
8 is a deficient number because (1 + 2 + 4) = 7
Prime numbers are deficient
Implement a way to determine whether a given number is perfect. Depending on your language track, you may also need to implement a way to determine whether a given number is abundant or deficient.

tests : `perfect-number.spec.js`

```
var PerfectNumbers = require('./perfect-numbers');

describe('Exercise - Perfect Numbers', function () {
  var perfectNumbers;

  beforeEach(function () {
    perfectNumbers = new PerfectNumbers();
  });

  describe('Perfect Numbers', function () {
    it('Smallest perfect number is classified correctly', function () {
      expect(perfectNumbers.classify(6)).toEqual('perfect');
    });
```

```
    it('Medium perfect number is classified correctly', function () {
      expect(perfectNumbers.classify(28)).toEqual('perfect');
    });

    it('Large perfect number is classified correctly', function () {
      expect(perfectNumbers.classify(33550336)).toEqual('perfect');
    });
  });

  describe('Abundant Numbers', function () {
    it('Smallest abundant number is classified correctly', function () {
      expect(perfectNumbers.classify(12)).toEqual('abundant');
    });

    it('Medium abundant number is classified correctly', function () {
      expect(perfectNumbers.classify(30)).toEqual('abundant');
    });

    it('Large abundant number is classified correctly', function () {
      expect(perfectNumbers.classify(33550335)).toEqual('abundant');
    });
  });

  describe('Deficient Numbers', function () {
    it('Smallest prime deficient number is classified correctly', function (
) {
      expect(perfectNumbers.classify(2)).toEqual('deficient');
    });

    it('Smallest non-prime deficient number is classified correctly', functi
on () {
      expect(perfectNumbers.classify(4)).toEqual('deficient');
    });

    it('Medium deficient number is classified correctly', function () {
      expect(perfectNumbers.classify(32)).toEqual('deficient');
    });

    it('Large deficient number is classified correctly', function () {
      expect(perfectNumbers.classify(33550337)).toEqual('deficient');
    });

    it('Edge case (no factors other than itself) is classified correctly', f
unction () {
      expect(perfectNumbers.classify(1)).toEqual('deficient');
    });
  });
```

```
  describe('Invalid Inputs', function () {
    it('Zero is rejected (not a natural number)', function () {
      expect(perfectNumbers.classify(0)).toEqual('Classification is only pos
sible for natural numbers.');
    });

    it('Negative integer is rejected (not a natural number)', function () {
      expect(perfectNumbers.classify(-1)).toEqual('Classification is only po
ssible for natural numbers.');
    });
  });
});
```