# Computer Science G12

## Term 2, Test 1. Tue Dec 19 2017

**Name :**

Note: Write your full name in capitals.

## Submission

Write your code in a text file (extension `txt`) and submitted as attachment via email.

## Problems

**Note**: The reference to the `print` function is the one available in the HTML Live Editor, which you can use.

1. Implement a counter object in Javascript such that the following code prints `32`. **Constraints**: The only instance properties are the methods `inc()` and getCount()‘.

   ```
   var i = new Counter(30)
   i.inc()
   i.inc()
   print("i:"+i)
   ```

2. You are provided a `matrix.js` script file and a `matrix-t2-test1.html` HTML test file. You'll need to modify them to satisfy the constraints stated below.

   Both files can be downloaded from http://msantos.sdf.org/G12/Term2 and are also attached in the appendix section below in this document.

   1. Implement the `toString` method such that the test file prints

      ```
      0 1
      1 0

      3
      3
      ```

   2. Modify the `toString` method you just wrote, as well as the Matrix constructor such that the following code

      ```
      var M = new Matrix([2,2],[0,1,1,0],"M")
      var v = new Matrix([2,1],[3,3],"v")
      ```

```
var nona = new Matrix([2,1],[5,7])

print(M)
print(v)
print(nona)
```

prints

```
M[ 0 1 ]
 [ 1 0 ]

v[ 3 ]
 [ 3 ]

noname[ 5 ]
      [ 7 ]
```

3. **Consistency test**: Implement a *consistency test* such that we get an error message printed in an alert and in the console saying `ERROR: matrix A : Incompatible data length (4) w/ matrix dimensions (2x3)` if we try to instantiate a matrix where the dimensions doesn't fit with the data length as, e.g.,

   ```
   var M = new Matrix([2,3],[0,1,1,0],"M")
   ```

4. **Transpose**: Implement the method `t` (lowercase 't') that returns the transpose of a matrix. Follow these examples:

   This code

   ```
   var A = new Matrix([2,3],[11,12,13,21,22,23],"A")
   var w = new Matrix([2,1],[3,3],"w")
   print(A)
   print(A.t())
   print(w)
   print(w.t())
   ```

   prints

   ```
   A[ 11 12 13 ]
    [ 21 22 23 ]

   A†[ 11 21 ]
     [ 12 22 ]
     [ 13 23 ]

   w[ 3 ]
    [ 3 ]

   w†[ 3 3 ]
   ```

where the symbol † (*dagger*) denotes the transposed matrix.

## Appendix

### Matrix HTML template file: matrix-t2-test1.html

```
<!doctype html>
<html>
  <!-- HTML template page: matrix-t2-test1.html
       http://msantos.sdf.org/G12/Term2/matrix2-t2-test1.html
  -->
  <head>
      <title>Matrix test bed</title>
      <meta charset="utf-8">
      <script>
        //Helper function
        function print(x){
          document.body.innerHTML += x + "<br>";
        }
      </script>
      <style>
        body { font-family: monospace ; }
      </style>
  </head>

  <body>


<script src="file://PATH/matrix.js"></script>
    <script>

var M = new Matrix([2,2],[0, 1, 1, 0])
var v = new Matrix([2,1],[3,3])
print(M)
print(v)



    </script>
  </body>
</html>
```

where `PATH` depends on wehter you run on Windows or Mac:

a. **Mac**: `PATH=/Users/'your-use-name'/Desktop`
b. **Windows**: `PATH=/C:/Users/'your-use-name'/Desktop`

and where the script file `matrix.js`, which can be downloaded from http://msantos.sdf.org/G12/Term2/, contains the matrix template library that can be found in the appendix below.

### Matrix Template Library: matrix.js

```
//Matrix Library

/*The matrix constructor

Input:
        dim :: array of 2 integers
        data:: array of all matrix elements from left->right and top->bottom

"Output": (not really a return value, mind you!)
        Object with properties:
            dim :: an array of 2 integers
            data:: a 1-dimensional array w/ actual elements sorted left->right,top->bottom
*/
function Matrix(dim=[0,0], data=[]){
        this.dim = dim ;
        this.data = data ;
}

Matrix.prototype.mult = function(B){
        if( this.dim[1] != B.dim[0] ) {
                var msg = "<p>ERROR: Can't multiply. #columns doesn't match #rows! </p>" ;
                document.body.innerHTML +=  msg ;
                console.log(msg);
                return -1;
        }

        var dim = [this.dim[0] , B.dim[1] ] ;
        var data = [] ;

        //
        for(var i = 0 ; i < this.dim[0] ; i++ ) {
                for(var j = 0 ; j < B.dim[1] ; j++) {
                        var sum=0;
                        for(var k = 0 ; k < this.dim[1] ; k++ ) {
                            sum += this.data[i*this.dim[1] + k] * B.data[k*B.dim[1] + j] ;
                        }
                        data.push( sum ) ;
                }
        }
```

```
        return new Matrix( dim, data ) ;
}

Matrix.prototype.toString = function(){

}
```