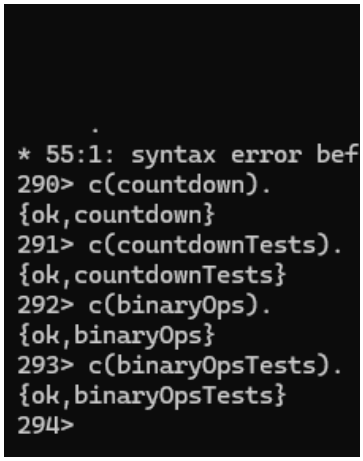
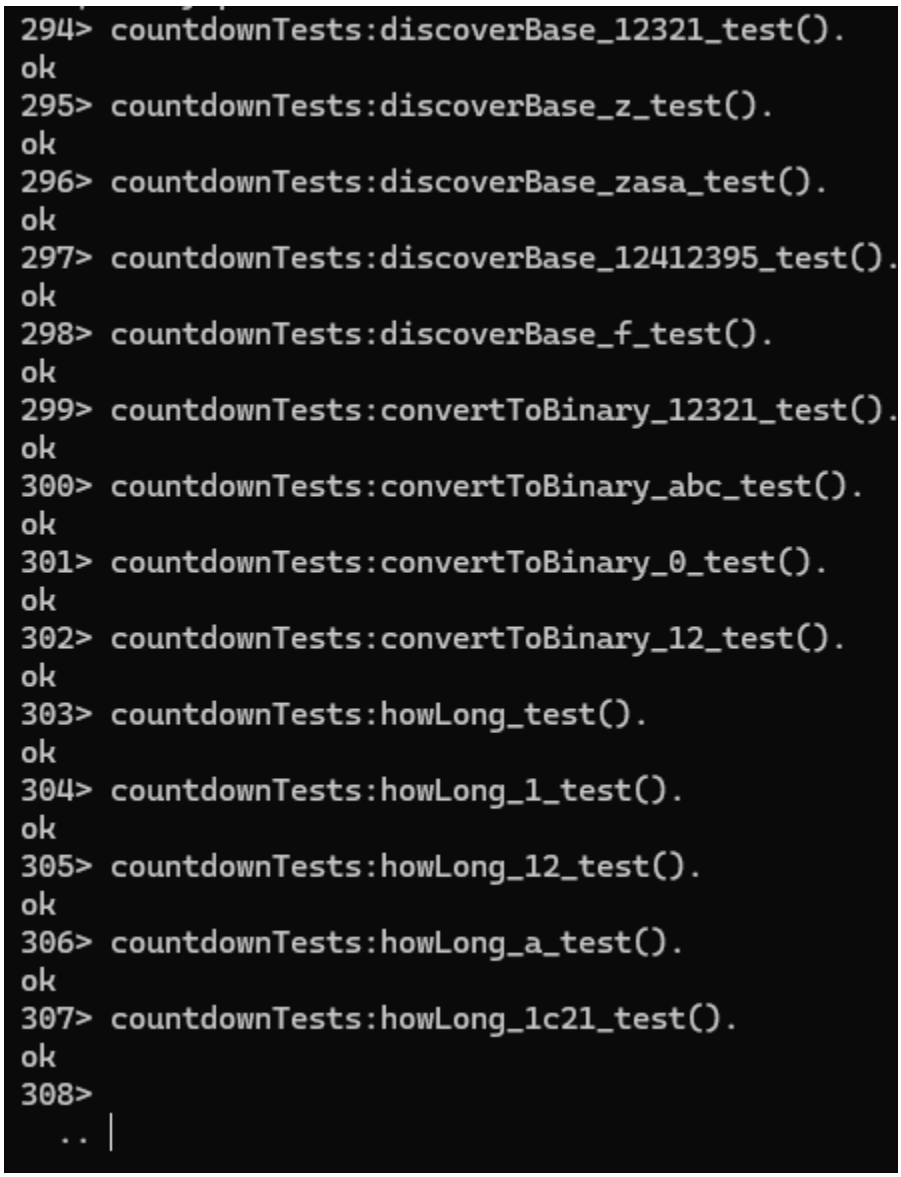
Compiling all files:



Running tests on countDown:



Running countdown:discoverBase

```
ok
308> countdown:discoverBase("12321").
4
309> countdown:discoverBase("z").
36
310> countdown:discoverBase("zasa").
36
311> countdown:discoverBase("12412395").
10
312> countdown:discoverBase("f").
16
```

Running countdown:generateBinaryConverter

```
323> f().
ok
324> F = countdown:generateBinaryConverter(4).
#Fun<countdown.0.8684977>
325> F("12321").
[1,0,0,1,1,1,0,1,1]
326> E = countdown:generateBinaryConverter(13).
#Fun<countdown.0.8684977>
327> E("abc").
[1,0,1,0,1,1,0,0,1,1,1]
328> G = countdown:generateBinaryConverter(1).
#Fun<countdown.0.8684977>
329> G("0").
[]
```

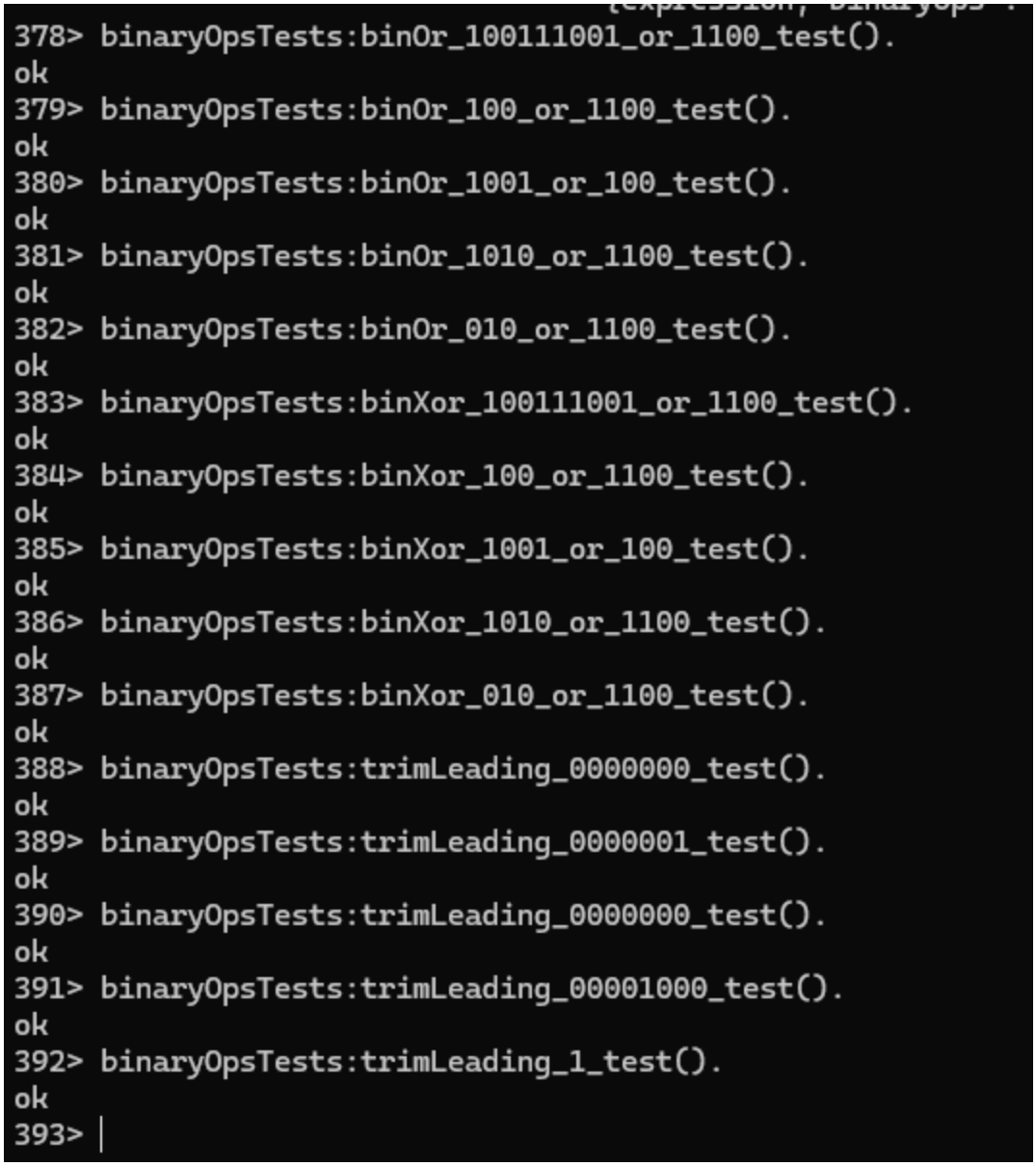
```
333> EI = countdown:generateBinaryConverter(3).
#Fun<countdown.0.8684977>
334> EI("12").
[1,0,1]
335>
```

Running countdown: howLongDoWeHave

```
340> countdown:howLongDoWeHave(["abc", "12321", "789", "1face", "z1"]).
[1,0,0,1,1,1,0,1,1]
341> countdown:howLongDoWeHave(["abc", "12321", "789", "1face", "z1"]).
[1,0,0,1,1,1,0,1,1]
342> countdown:howLongDoWeHave(["1", "12z"]).
[1]
343> countdown:howLongDoWeHave(["12"]).
[1,0,1]
344> countdown:howLongDoWeHave(["a"]).
[0,1,0,1]
345> countdown:howLongDoWeHave(["1c21"]).
[0,0,1,1,1,0,0,1,0,0,0,0,1]
346>
```

Running tests on binaryOpsTests:

```
368> binaryOpsTests:binNegate_100111011_test().
ok
369> binaryOpsTests:binNegate_000111011_test().
ok
370> binaryOpsTests:binNegate_111111111_test().
ok
371> binaryOpsTests:binNegate_000000000_test().
ok
372> binaryOpsTests:binAnd_100111001_and_1100_test().
ok
373> binaryOpsTests:binAnd_1_and_1_test().
ok
374> binaryOpsTests:binAnd_0_and_0_test().
** exception error: {assertEqual,[{module,binaryOpsTests},
                                  {line,21},
                                  {expression,"binaryOps : binAnd ( [ 0 ] , [ 0 ] )"},
                                  {expected,[]},
                                  {value,[0]}]}
     in function  binaryOpsTests:binAnd_0_and_0_test/0 (binaryOpsTests.erl, line 21)
375> binaryOpsTests:binAnd_011_and_110_test().
** exception error: {assertEqual,[{module,binaryOpsTests},
                                  {line,23},
                                  {expression,"binaryOps : binAnd ( [ 0 , 1 , 1 ] , [ 1 , 1 , 0 ] )"},
                                  {expected,[1]},
                                  {value,[0,1,0]}]}
     in function  binaryOpsTests:binAnd_011_and_110_test/0 (binaryOpsTests.erl, line 23)
```



Running binaryOps

```
346> binaryOps:binNegate([1,0,0,1,1,1,0,1,1]).
[1,1,0,0,0,1,0,0]
347> binaryOps:binNegate([0,0,0,1,1,1,0,1,1]).
[1,1,1,0,0,0,1,0,0]
348> binaryOps:binNegate([1,1,1,1,1,1,1,1,1]).
[]
349> binaryOps:binNegate([0,0,0,0,0,0,0,0,0]).
[1,1,1,1,1,1,1,1,1]
350> binaryOps:binAnd([1,0,0,1,1,1,0,0,1], [1,1,0,0]).
[1,0,0,0]
351> binaryOps:binAnd([1], [1]).
[1]
352> binaryOps:binAnd([0], [0]).
[0]
353> binaryOps:binAnd([0,1,1], [1,1,0]).
[0,1,0]
354> binaryOps:binOr([1,0,0,1,1,1,0,0,1], [1,1,0,0]).
[1,1,0,1]
355> binaryOps:binOr([1,0,0], [1,1,0,0]).
[1,1,0]
356> binaryOps:binOr([1,0,0,1], [1,0,0]).
[1,0,0]
357> binaryOps:binOr([1,0,1,0], [1,1,0,0]).
[1,1,1,0]
358> binaryOps:binOr([0,1,0], [1,1,0,0]).
[1,1,0]
359> binaryOps:binXor([1,0,0,1,1,1,0,0,1], [1,1,0,0]).
[0,1,0,1]
360> binaryOps:binXor([1,0,0], [1,1,0,0]).
[0,1,0]
361> binaryOps:binXor([1,0,0,1], [1,0,0]).
[0,0,0]
362> binaryOps:binXor([1,0,1,0], [1,1,0,0]).
[0,1,1,0]
363> binaryOps:binXor([0,1,0], [1,1,0,0]).
[1,0,0]
364> binaryOps:trimLeading([0,0,0,0,0,0,0]).
[]
365> binaryOps:trimLeading([0,0,0,0,0,0,1]).
[1]
366> binaryOps:trimLeading([0,0,0,0,1,0,0,0]).
[1,0,0,0]
367> binaryOps:trimLeading([1]).
[1]
368>
```