Excercise 11

11.1.1

To print the number of lines in the file we used below method:

```
println(readLines().count())
```

This gives us 10849 lines.

11.1.2

We used the given function tweetFromLine() to get wel formed tweets. If the tweet is not well formted, we simply filter it away.

```
fun getWelformedTweetsAsStream(): Stream<Tweet> {
    var linesStream = readLines().asStream()
    var myStream = linesStream
        .filter({ p -> tweetFromLine(p) != null})
        .map { p -> tweetFromLine(p)!! }
    return myStream;
}
```

11.1.3

We a tweet channel which the producer sends messages to, and the consumer receives from. The channel is buffered with a size of 200 to ensure that the messages are send for every 200 messages.

```
val tweetChannel = Channel<Tweet>(200)

suspend fun consumer(){
    while (true)
        println(tweetChannel.receive())
}

suspend fun producer(){
    var iter = getWelformedTweetsAsStream().iterator()

    while(iter.hasNext())
    {
        tweetChannel.send(iter.next())
        sleep(5)
    }
}
```

And to start the producer and consumer in a blocking coroutine:

```
runBlocking {
    launch { producer() }
    launch { consumer() }
 }
```

11.1.4

Here the consumer listens for all tweet coming in from the channel, filters it based on airline and send the message to another channel that is directed to a specific airline.

```
val tweetChannel2 = Channel<Tweet>()
val tweetChannelAmerican = Channel<Tweet>(25)
val tweetChannelSouthwest = Channel<Tweet>(25)

suspend fun producer2(){
    var iter = getWelformedTweetsAsStream().iterator()

    while(iter.hasNext())
    {
        tweetChannel2.send(iter.next())
    }
}


suspend fun consumer2(){
    while (true){
        var tweet = tweetChannel2.receive();
        if(tweet.airline == "American"){
            tweetChannelAmerican.send(tweet)
        }
        else if(tweet.airline == "Southwest"){
            tweetChannelSouthwest.send(tweet)
        }
    }
}

suspend fun americanConsumer(){
    while (true)
        println(tweetChannelAmerican.receive())
}

suspend fun southwestConsumer(){
    while (true)
        println(tweetChannelSouthwest.receive())
}
```

11.1.5

Here, additional filtering is applied for the consumer that listens for messages for a specific airline so only messages that meet a specific requirement are accounted for.

```
val tweetChannel3 = Channel<Tweet>()
val tweetChannelAmerican2 = Channel<Tweet>(25)
val tweetChannelSouthwest2 = Channel<Tweet>(25)

suspend fun producer3(){
    var iter = getWelformedTweetsAsStream().iterator()

    while(iter.hasNext())
    {
        tweetChannel3.send(iter.next())
    }
}

suspend fun consumer3(){
    while (true){
        var tweet = tweetChannel3.receive();
        if(tweet.airline == "AmericanAir"){
            tweetChannelAmerican2.send(tweet)
        }
        else if(tweet.airline == "Southwest"){
            tweetChannelSouthwest2.send(tweet)
        }
    }
}

suspend fun americanConsumer2(){
    while (true)
    {
        var tweet = tweetChannelAmerican2.receive()
        if(tweet.negativereason_confidence != null &&
tweet.negativereason_confidence.toDouble() > 0.75)
            println(tweetChannelAmerican2.receive())
    }
}

suspend fun southwestConsumer2(){
    while (true)
    {
        var tweet = tweetChannelSouthwest2.receive()
        if(tweet.negativereason != null && tweet.negativereason != "")
            println(tweetChannelAmerican2.receive())
    }
}
```

11.2.1

We think that the print will affect the time, as the thread will be occupied for a longer period of time and not released for other work to take the thread.
The number of cores does not influence the time it takes to create a thread, however it would influence the number of threads that is possible to run and therefore, if running more threads than cores in the computer, some threads will have to wait until threads are released by the code.

11.2.2

In our result it is evident that routines are actually slower to start up than threads are.
Creating threads in java takes approx. 180.345 ns, while in kotlin coroutines take approx. 124.384.616

3) The results that I obtained from running both snippets of code were:

CODE I	CODE II
Starttime: 8,227,379	Starttime: 9207209
Starttime: 26,442,698	Starttime: 6748617
Starttime: 23,689,760	Starttime: 4223754
Starttime: 23,772,087	Starttime: 4238807
Starttime: 23,850,471	Starttime: 4253504
Starttime: 23,932,347	Starttime: 4316864
Starttime: 24,011,855	Starttime: 4334148
Starttime: 23,974,130	Starttime: 4266867
Starttime: 24,196,018	Starttime: 4274148
Starttime: 24,293,277	Starttime: 4324535
Starttime: 24,360,648	Starttime: 4368431
Starttime: 24,423,469	Starttime: 4403315
Starttime: 24,507,850	Starttime: 4422342
Starttime: 24,615,203	Starttime: 4441814
Starttime: 24,711,370	Starttime: 4444228
Starttime: 24,767,371	Starttime: 4431860
Starttime: 24,858,748	Starttime: 4449281
Starttime: 24,941,250	Starttime: 4473773
Starttime: 25,024,128	Starttime: 4489705
Starttime: 25,105,824	Starttime: 4507116
Starttime: 25,305,069	Starttime: 4524254
Starttime: 25,384,668	Starttime: 4541618
Starttime: 25,506,272	Starttime: 4562195
Starttime: 25,559,790	Starttime: 4582945
Starttime: 25,619,736	Starttime: 4602730
Starttime: 25,702,160	Starttime: 4623470
Starttime: 25,860,593	Starttime: 4645595
Starttime: 25,989,956	Starttime: 4680408
Starttime: 26,091,086	Starttime: 4723622
Starttime: 26,202,078	Starttime: 4731216
Starttime: 26,279,079	Starttime: 4734299

Starttime: 26,291,561	Starttime: 4659930
Starttime: 26,467,585	Starttime: 4735494
Starttime: 26,545,168	Starttime: 4756276
Starttime: 26,623,819	Starttime: 4810202
Starttime: 26,701,064	Starttime: 4828214
Starttime: 26,776,811	Starttime: 4846697
Starttime: 27,021,430	Starttime: 4864593
Starttime: 27,106,346	Starttime: 4886808
Starttime: 27,185,470	Starttime: 4934357
Starttime: 27,267,398	Starttime: 4971590
Starttime: 27,361,744	Starttime: 5012733
Starttime: 27,529,410	Starttime: 5088678
Starttime: 27,664,178	Starttime: 5114036
Starttime: 27,741,701	Starttime: 5102174
Starttime: 27,827,643	Starttime: 5143915
Starttime: 27,901,372	Starttime: 5163592
Starttime: 27,982,214	Starttime: 5182420
Starttime: 28,056,621	Starttime: 5255515
Starttime: 28,136,689	Starttime: 5299610
Starttime: 28,213,243	Starttime: 5340621
Starttime: 28,286,224	Starttime: 5383662
Starttime: 28,418,587	Starttime: 5439710
Starttime: 28,509,654	Starttime: 5495292
Starttime: 28,671,873	Starttime: 5554620
Starttime: 28,793,246	Starttime: 5615826
Starttime: 28,894,969	Starttime: 5642084
Starttime: 28,974,984	Starttime: 5659729
Starttime: 29,152,769	Starttime: 5678150
Starttime: 29,269,264	Starttime: 5703942
Starttime: 29,341,415	Starttime: 5728678
Starttime: 29,418,339	Starttime: 5752316
Starttime: 29,591,845	Starttime: 5771615
Starttime: 29,631,770	Starttime: 5629050
Starttime: 29,730,583	Starttime: 5652523
Starttime: 29,809,311	Starttime: 5646666

Starttime: 29,893,985	Starttime: 5650154
Starttime: 29,992,763	Starttime: 5671350
Starttime: 30,074,256	Starttime: 5694716
Starttime: 30,166,574	Starttime: 5692261
Starttime: 30,242,348	Starttime: 5693592
Starttime: 30,319,621	Starttime: 5714376
Starttime: 30,394,111	Starttime: 5710114
Starttime: 30,471,667	Starttime: 5730523
Starttime: 30,545,500	Starttime: 5759988
Starttime: 30,621,837	Starttime: 5799542
Starttime: 30,696,141	Starttime: 5803458
Starttime: 30,791,343	Starttime: 5854872
Starttime: 30,882,413	Starttime: 5916951
Starttime: 30,967,085	Starttime: 5968218
Starttime: 31,049,001	Starttime: 5990989
Starttime: 31,124,480	Starttime: 6012784
Starttime: 31,210,002	Starttime: 6057647
Starttime: 31,289,948	Starttime: 6093909
Starttime: 31,376,691	Starttime: 6416180
Starttime: 31,455,241	Starttime: 6402816
Starttime: 31,545,688	Starttime: 6435938
Starttime: 31,641,475	Starttime: 6442113
Starttime: 31,737,363	Starttime: 6509951
Starttime: 31,820,631	Starttime: 6554502
Starttime: 31,900,370	Starttime: 6620778
Starttime: 31,997,496	Starttime: 6661387
Starttime: 32,092,321	Starttime: 6695348
Starttime: 32,177,689	Starttime: 6709157
Starttime: 32,258,053	Starttime: 6730605
Starttime: 32,344,083	Starttime: 6754700
Starttime: 32,432,109	Starttime: 6788659
Starttime: 32,530,413	Starttime: 6881060
Starttime: 32,610,237	Starttime: 6887832
Starttime: 32,667,564	Starttime: 6891211

It can therefore be observed that the differences in start-up times between both programs is indeed about a factor of 4.

4)

CODE - UNCONFINED	CODE - DEFAULT
Starttime: 2464094	Starttime: 17210062
Starttime: 48069	Starttime: 2188268
Starttime: 23210	Starttime: 1230116
Starttime: 20852	Starttime: 95840
Starttime: 36018	Starttime: 60081
Starttime: 47770	Starttime: 69401
Starttime: 20450	Starttime: 85443
Starttime: 23555	Starttime: 59789
Starttime: 36317	Starttime: 66204
Starttime: 37000	Starttime: 55434
Starttime: 19951	Starttime: 49202
Starttime: 19404	Starttime: 61004
Starttime: 18839	Starttime: 55297
Starttime: 19159	Starttime: 46819
Starttime: 18906	Starttime: 59098
Starttime: 38402	Starttime: 40119
Starttime: 19008	Starttime: 53605
Starttime: 19152	Starttime: 70792
Starttime: 55973	Starttime: 47354
Starttime: 36516	Starttime: 42314
Starttime: 37026	Starttime: 49943
Starttime: 42786	Starttime: 59192
Starttime: 18496	Starttime: 46142
Starttime: 38371	Starttime: 83376
Starttime: 40644	Starttime: 60613
Starttime: 42989	Starttime: 61120
Starttime: 57493	Starttime: 54573
Starttime: 18485	Starttime: 65214
Starttime: 18837	Starttime: 58683
Starttime: 37091	Starttime: 60503

Starttime: 18772	Starttime: 51607
Starttime: 66676	Starttime: 51647
Starttime: 20244	Starttime: 43871
Starttime: 23907	Starttime: 52194
Starttime: 24607	Starttime: 46353
Starttime: 21024	Starttime: 49608
Starttime: 19772	Starttime: 61313
Starttime: 20639	Starttime: 56049
Starttime: 19458	Starttime: 51314
Starttime: 23185	Starttime: 59198
Starttime: 19155	Starttime: 68411
Starttime: 23578	Starttime: 90935
Starttime: 142446	Starttime: 82592
Starttime: 31811	Starttime: 47258
Starttime: 21656	Starttime: 52936
Starttime: 22122	Starttime: 59724
Starttime: 36578	Starttime: 85495
Starttime: 38508	Starttime: 83704
Starttime: 18621	Starttime: 69040
Starttime: 18712	Starttime: 58107
Starttime: 48941	Starttime: 61263
Starttime: 23865	Starttime: 67181
Starttime: 20442	Starttime: 46459
Starttime: 38050	Starttime: 54190
Starttime: 18927	Starttime: 69229
Starttime: 23350	Starttime: 35662
Starttime: 37757	Starttime: 36852
Starttime: 35335	Starttime: 51813
Starttime: 17467	Starttime: 55935
Starttime: 51015	Starttime: 49308
Starttime: 47573	Starttime: 66053
Starttime: 35618	Starttime: 77776
Starttime: 41636	Starttime: 91561
Starttime: 101671	Starttime: 38530
Starttime: 88219	Starttime: 48069

Starttime: 23591	Starttime: 45184
Starttime: 49721	Starttime: 62873
Starttime: 19308	Starttime: 74155
Starttime: 41499	Starttime: 70422
Starttime: 34246	Starttime: 48474
Starttime: 35761	Starttime: 86250
Starttime: 19603	Starttime: 64562
Starttime: 50719	Starttime: 66204
Starttime: 47114	Starttime: 57521
Starttime: 17242	Starttime: 60500
Starttime: 17395	Starttime: 43854
Starttime: 17467	Starttime: 64760
Starttime: 18084	Starttime: 75451
Starttime: 17081	Starttime: 86610
Starttime: 19543	Starttime: 81776
Starttime: 49429	Starttime: 59701
Starttime: 17204	Starttime: 69305
Starttime: 17145	Starttime: 80109
Starttime: 17003	Starttime: 70178
Starttime: 53012	Starttime: 48568
Starttime: 18770	Starttime: 63108
Starttime: 18787	Starttime: 41428
Starttime: 21200	Starttime: 67161
Starttime: 22068	Starttime: 50741
Starttime: 31263	Starttime: 659132
Starttime: 35044	Starttime: 72111
Starttime: 17440	Starttime: 149327
Starttime: 17652	Starttime: 45559
Starttime: 17631	Starttime: 50607
Starttime: 45631	Starttime: 50975
Starttime: 65812	Starttime: 53959
Starttime: 17536	Starttime: 45603
Starttime: 16496	Starttime: 46791
Starttime: 16621	Starttime: 61669
Starttime: 33337	Starttime: 67124

3)

```
import kotlinx.coroutines.*
import kotlinx.coroutines.channels.*

class tryout {
    companion object {
        @JvmStatic
        fun main(args: Array<String>) {
            tryout3()

        }

        fun tryout3():Unit {
            runBlocking { this: CoroutineScope
                val measurement = Channel<Long>()
                val reps = 100
                for(i in 0..reps-1){
                    val start = System.nanoTime()
                    launch { this: CoroutineScope
                        val time = System.nanoTime() - start
                        measurement.send(time)
                    }
                }
                launch{ this: CoroutineScope
                    measurementCollector(reps, measurement)
                } ^runBlocking
            }
        }
```

```
    suspend fun measurementCollector(n: Int, measurements: Channel<Long>): Unit {
        var count = 0L
        var sum = 0L
        var ssm = 0L
        // *** TO DO ***
        while(count < n){
            val receivedValue = measurements.receive()
            sum += receivedValue
            ssm += receivedValue*receivedValue
            count += 1
        }
        //Receive the measurements, add the measurements and determine the mean
        val mean = (sum/count).toDouble()
        val sdev = Math.sqrt( (ssm - mean*mean*count).toDouble() /(count-1))
        println("%,6.1f ns +/- %,8.2f %,d%n".format( mean, sdev, count) )
    }
```