1.
    a. To set up the DStream, I did the following steps:

In the first terminal, open pyspark (using the command `pyspark` in the right folder), then put the following codes in:

```
sc.stop()
from pyspark import SparkContext
from pyspark.streaming import StreamingContext
from pyspark import StorageLevel

sc = SparkContext("local[2]", "StreamingreduceByWindow")
ssc = StreamingContext(sc, 1)
lines = ssc.socketTextStream("localhost", 9999, StorageLevel.MEMORY_AND_DISK)
input.pprint()
ssc.start()
ssc.awaitTermination()
```

In the second terminal, use the command `nc -lk 9999`

In this way, input read from the second terminal will be streamed into the first terminal, where the program receives and does the moving average calculations. The output will get emitted in the second terminal as well.

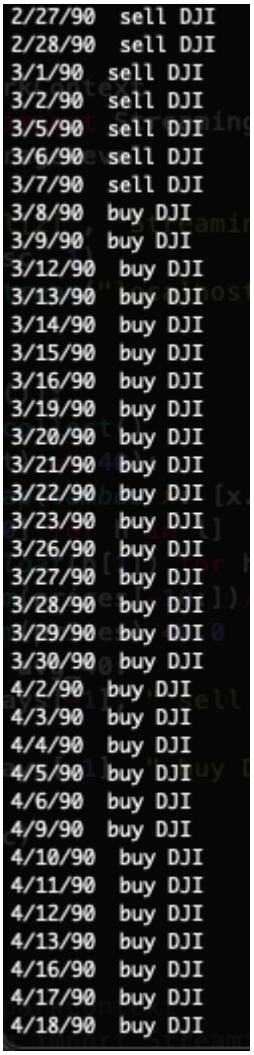
    b. Combined with c
    c. The codes I set up to calculate the moving average:

```
lines = ssc.socketTextStream("localhost", 9999,
StorageLevel.MEMORY_AND_DISK).window(40, 1)

def my_func(rdd):
    if (not rdd.isEmpty()):
        rdd_list = rdd.collect()
        if (len(rdd_list) == 40):
            l =  list(map(lambda x: [x.split(" ")[0], x.split(" ")[1]],
rdd_list))
            days = [ h[0] for h in l]
            prices = [float(h[1]) for h in l]
            avg_10 = sum(prices[-10:])/10.0
            avg_40 = sum(prices)/40.0
            if avg_10 < avg_40:
                print(days[-1], " sell DJI")
            else:
                print(days[-1], " buy DJI")

lines.foreachRDD(my_func)
```

I reset the wait time in dj30-feeder to be 1s, and used the window function to get a rolling window of 40 days every day. After passing the streams into the helper function using foreachRDD, I make sure to check the length of the rdd stream (should be equal to 40) so as to discard the first 40 days when MA40 can't be calculated. After both moving averages are

calculated, I checked to decide whether it's a buy or a sell for that specific day. The output is then emitted (printed) to the screen.

d. The sample output for this task:



2.

a. The program I used to create bloom filter:

```
filter_size = len(understood_words)*8
bit_array = bitarray(filter_size)
bit_array.setall(0)
def add(bit_array, item):
    digests = []
    for i in range(10):
        digest = mmh3.hash(item,i, signed=False) % filter_size
        digests.append(digest)
        bit_array[digest] = True

for word in understood_words:
    add(bit_array, word)

with open('bloom.txt', 'wb') as bloom:
    bit_array.tofile(bloom)
```

b. Pyspark code:

```
sc.stop()
from pyspark import SparkContext
from pyspark.streaming import StreamingContext
from pyspark.streaming import DStream
from pyspark import StorageLevel

import mmh3
import re
import os
from bitarray import bitarray

os.environ['PYSPARK_PYTHON'] = "./pyspark_pex_env.pex"

def pre_process(text):
    text=text.lower()
    text=re.sub("","",text)
    text = re.sub(r'[0-9\,$]+', '', text)
    text=re.sub("(\\d|\\W)+"," ",text)
    text = ' '.join([word for word in text.split() if len(word) > 3])
    return text

def check(line):
    dir_path = os.path.dirname(os.path.realpath('bloom.txt'))
    bit_array = bitarray()
    with open(dir_path + '/bloom.txt', 'rb') as bloom:
        bit_array.fromfile(bloom)
    for i in range(10):
        for word in line:
            word = pre_process(word)
            digest = mmh3.hash(word,1) % 169440
            # if any of bit is False then, it's not present in the filter
            # (unfamiliar), so we should keep the title (return True)
            if bit_array[digest] == False:
                return True
    return False

sc = SparkContext("local[2]", "Bloom2")
ssc = StreamingContext(sc, 1)
lines = ssc.socketTextStream("localhost", 9999, StorageLevel.MEMORY_AND_DISK)
words = lines.map(lambda line: line.split(" ")).filter(lambda x: check(x))
words = words.map(lambda list_line: ' '.join(list_line))
words.pprint()
ssc.start()
ssc.awaitTermination(5)
```

c. My DEMO recording: https://tufts.zoom.us/rec/share/gU2nE884waEsNoxUPdujmH0GqCZWBMcSLPKFBTooFMbDNGmLtrg4c6ALWqrjilUp.EWlg0k8VIeXnYBfq?startTime=1649634446000