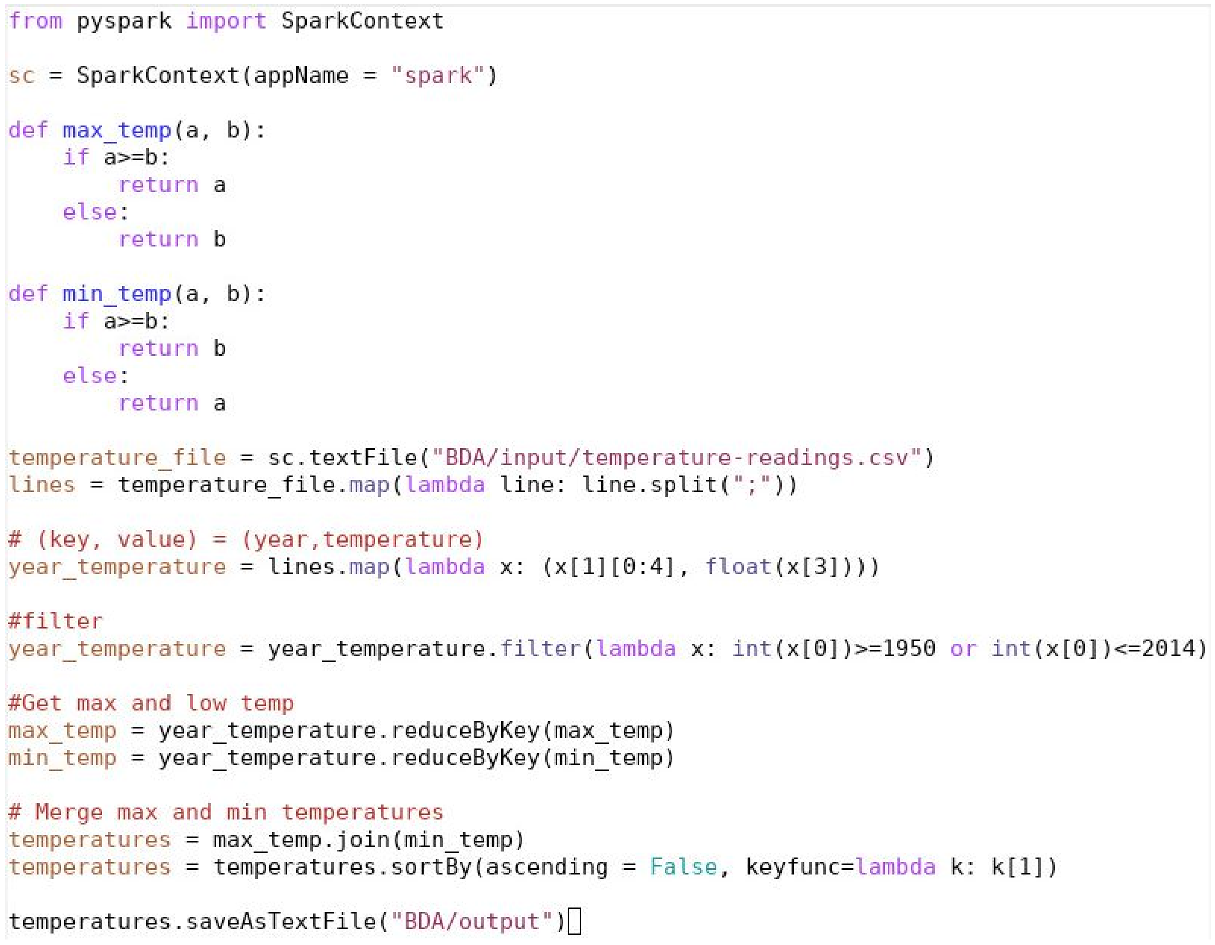
Alice Velander, alive213
David Björelind, davbj395

TDDE31: Big Data Analysis - Lab 1

Generally, if more than 15 rows was obtained in the output, only the first 15 is given in the answers.

1)
Code:



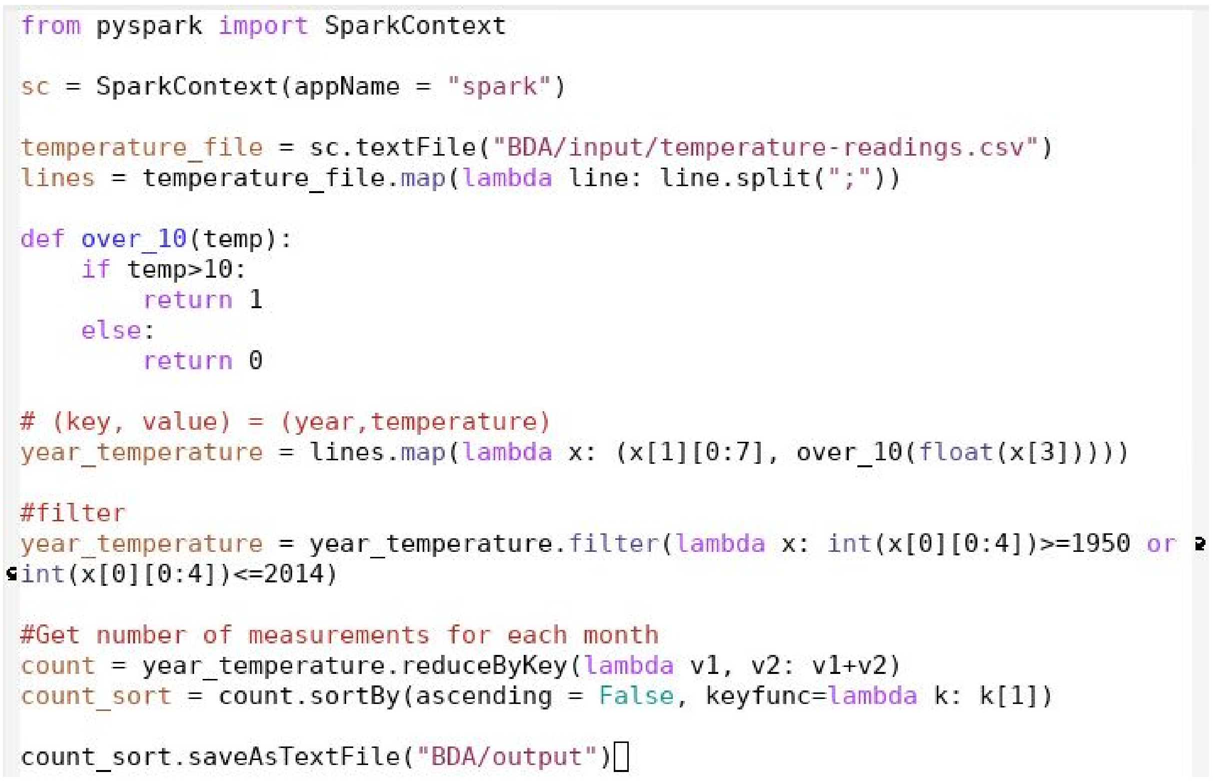
```
from pyspark import SparkContext

sc = SparkContext(appName = "spark")

def max_temp(a, b):
    if a>=b:
        return a
    else:
        return b

def min_temp(a, b):
    if a>=b:
        return b
    else:
        return a

temperature_file = sc.textFile("BDA/input/temperature-readings.csv")
lines = temperature_file.map(lambda line: line.split(";"))

# (key, value) = (year,temperature)
year_temperature = lines.map(lambda x: (x[1][0:4], float(x[3])))

#filter
year_temperature = year_temperature.filter(lambda x: int(x[0])>=1950 or int(x[0])<=2014)

#Get max and low temp
max_temp = year_temperature.reduceByKey(max_temp)
min_temp = year_temperature.reduceByKey(min_temp)

# Merge max and min temperatures
temperatures = max_temp.join(min_temp)
temperatures = temperatures.sortBy(ascending = False, keyfunc=lambda k: k[1])

temperatures.saveAsTextFile("BDA/output")
```

```
========================== FINAL OUTPUT ==========================
(u'1975', (36.1, -37.0))
(u'1992', (35.4, -36.1))
(u'1994', (34.7, -40.5))
(u'2010', (34.4, -41.7))
(u'2014', (34.4, -42.5))
(u'1947', (34.3, -32.0))
(u'1989', (33.9, -38.2))
(u'1982', (33.8, -42.2))
(u'1968', (33.7, -42.0))
(u'1966', (33.5, -49.4))
(u'1945', (33.4, -26.3))
(u'1983', (33.3, -38.2))
```

Alice Velander, alive213
David Björelind, davbj395

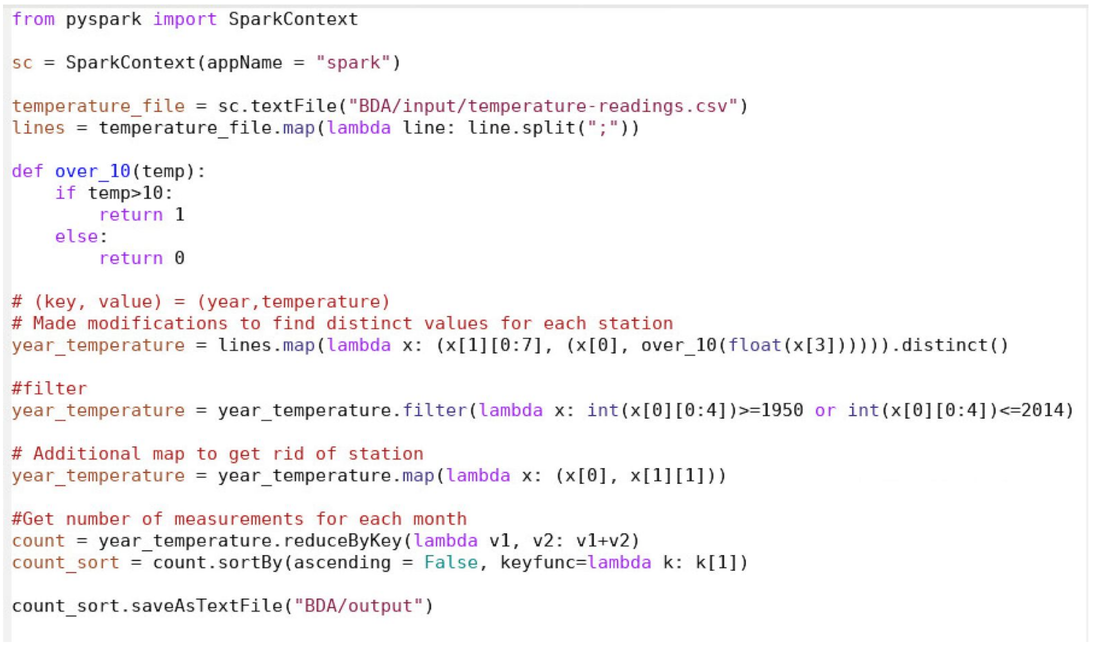
```
(u'2002', (33.3, -42.2))
(u'1970', (33.2, -39.6))
(u'1986', (33.2, -44.2))
```

2)
Code:
i)



```python
from pyspark import SparkContext

sc = SparkContext(appName = "spark")

temperature_file = sc.textFile("BDA/input/temperature-readings.csv")
lines = temperature_file.map(lambda line: line.split(";"))

def over_10(temp):
    if temp>10:
        return 1
    else:
        return 0

# (key, value) = (year,temperature)
year_temperature = lines.map(lambda x: (x[1][0:7], over_10(float(x[3]))))

#filter
year_temperature = year_temperature.filter(lambda x: int(x[0][0:4])>=1950 or int(x[0][0:4])<=2014)

#Get number of measurements for each month
count = year_temperature.reduceByKey(lambda v1, v2: v1+v2)
count_sort = count.sortBy(ascending = False, keyfunc=lambda k: k[1])

count_sort.saveAsTextFile("BDA/output")
```

ii)
Made some changes to the code to take the distinct constraint into consideration:



```python
from pyspark import SparkContext

sc = SparkContext(appName = "spark")

temperature_file = sc.textFile("BDA/input/temperature-readings.csv")
lines = temperature_file.map(lambda line: line.split(";"))

def over_10(temp):
    if temp>10:
        return 1
    else:
        return 0

# (key, value) = (year,temperature)
# Made modifications to find distinct values for each station
year_temperature = lines.map(lambda x: (x[1][0:7], (x[0], over_10(float(x[3]))))).distinct()

#filter
year_temperature = year_temperature.filter(lambda x: int(x[0][0:4])>=1950 or int(x[0][0:4])<=2014)

# Additional map to get rid of station
year_temperature = year_temperature.map(lambda x: (x[0], x[1][1]))

#Get number of measurements for each month
count = year_temperature.reduceByKey(lambda v1, v2: v1+v2)
count_sort = count.sortBy(ascending = False, keyfunc=lambda k: k[1])

count_sort.saveAsTextFile("BDA/output")
```

========================== FINAL OUTPUT ==========================

Alice Velander, alive213
David Björelind, davbj395

i)

```
(u'2014-07', 147681)
(u'2011-07', 146656)
(u'2010-07', 143419)
(u'2012-07', 137477)
(u'2013-07', 133657)
(u'2009-07', 133008)
(u'2011-08', 132734)
(u'2009-08', 128349)
(u'2013-08', 128235)
(u'2003-07', 128133)
(u'2002-07', 127956)
(u'2006-08', 127622)
(u'2008-07', 126973)
(u'2015-08', 126260)
(u'2002-08', 126073)
(u'2005-07', 125294)
(u'2011-06', 125193)
(u'2012-08', 125037)
(u'2006-07', 124794)
(u'2010-08', 124417)
(u'2014-08', 124045)
(u'1997-07', 123496)
(u'2007-07', 123218)
```

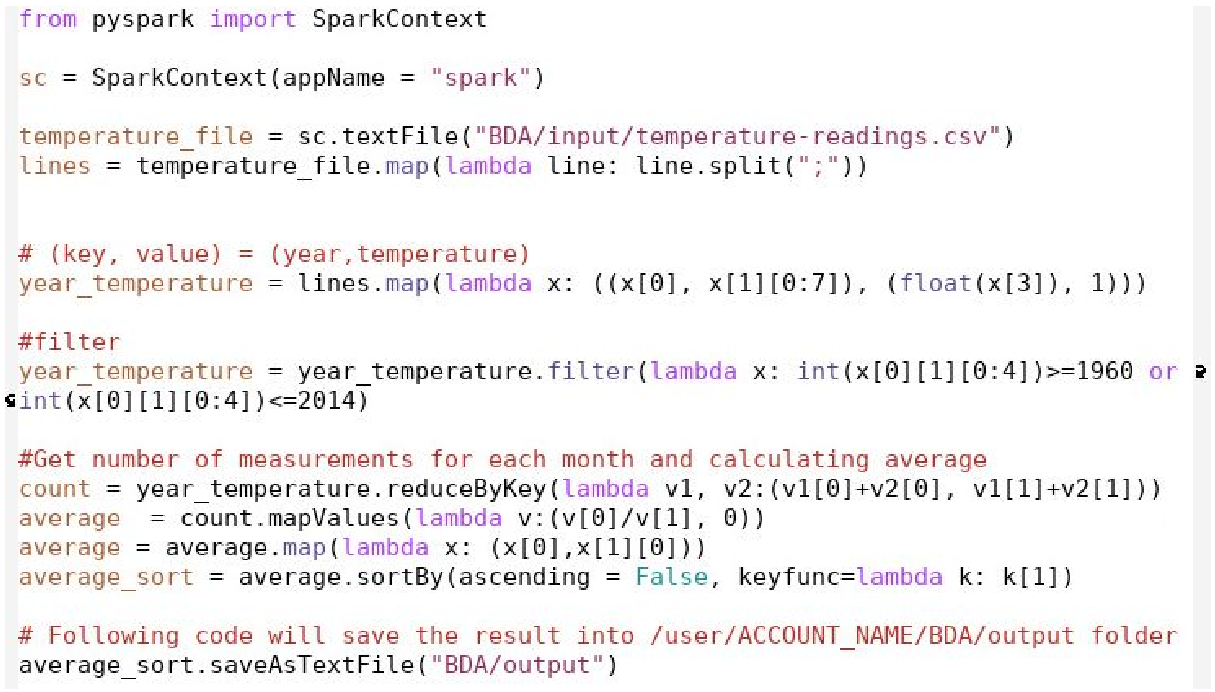
ii)

```
(u'1972-10', 378)
(u'1973-05', 377)
(u'1973-06', 377)
(u'1973-09', 376)
(u'1972-08', 376)
(u'1972-06', 375)
(u'1972-09', 375)
(u'1972-05', 375)
(u'1971-08', 375)
(u'1971-09', 374)
(u'1971-06', 374)
(u'1972-07', 374)
(u'1973-08', 373)
(u'1971-05', 373)
(u'1974-06', 372)
(u'1974-08', 372)
(u'1974-05', 370)
```

Alice Velander, alive213
David Björelind, davbj395

```
(u'1970-08', 370)
(u'1973-07', 370)
(u'1971-07', 370)
(u'1974-09', 370)
(u'1975-09', 369)
(u'1970-09', 369)
```

3)
Code:

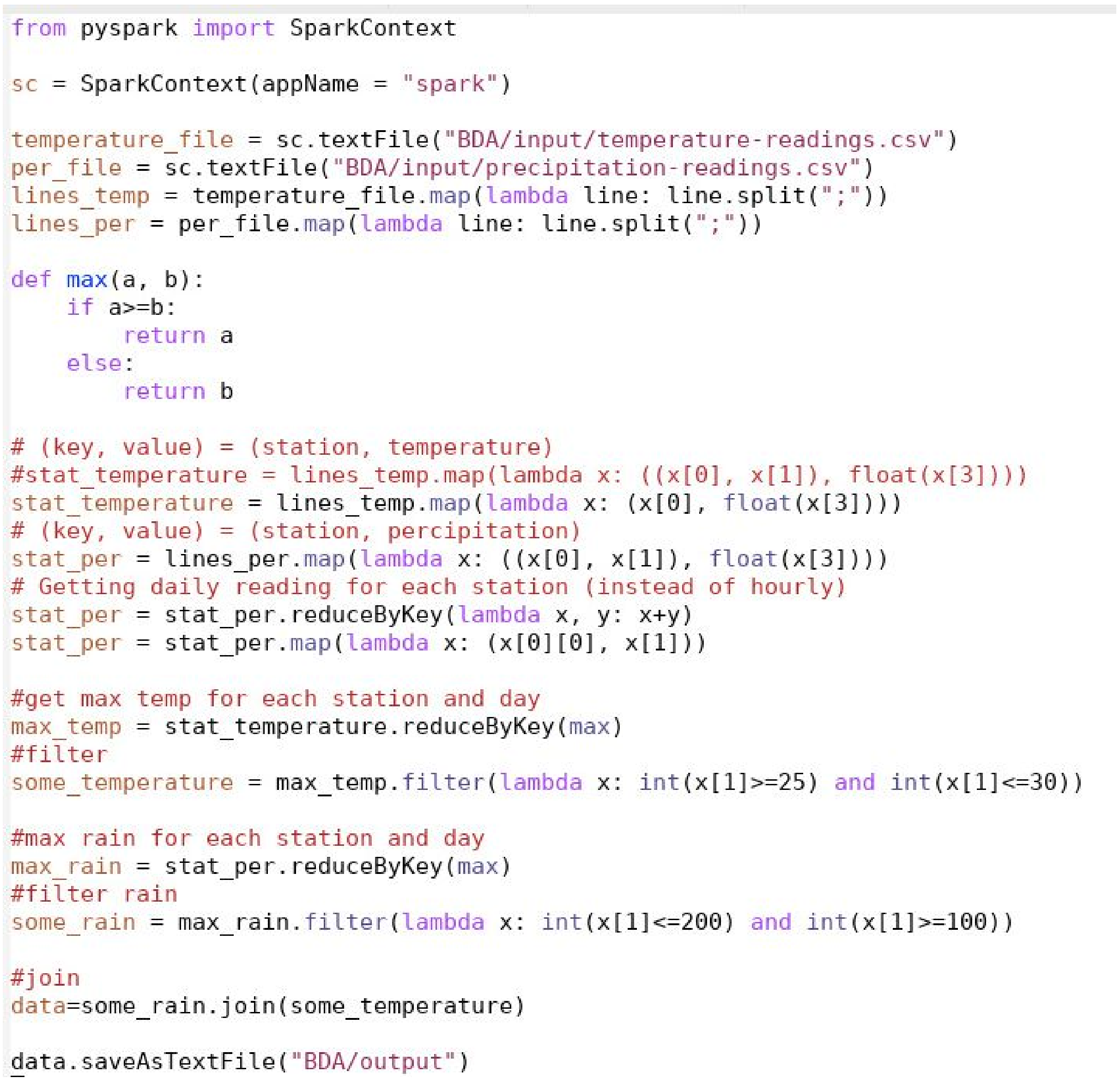


```python
from pyspark import SparkContext

sc = SparkContext(appName = "spark")

temperature_file = sc.textFile("BDA/input/temperature-readings.csv")
lines = temperature_file.map(lambda line: line.split(";"))


# (key, value) = (year,temperature)
year_temperature = lines.map(lambda x: ((x[0], x[1][0:7]), (float(x[3]), 1)))

#filter
year_temperature = year_temperature.filter(lambda x: int(x[0][1][0:4])>=1960 or int(x[0][1][0:4])<=2014)

#Get number of measurements for each month and calculating average
count = year_temperature.reduceByKey(lambda v1, v2:(v1[0]+v2[0], v1[1]+v2[1]))
average  = count.mapValues(lambda v:(v[0]/v[1], 0))
average = average.map(lambda x: (x[0],x[1][0]))
average_sort = average.sortBy(ascending = False, keyfunc=lambda k: k[1])

# Following code will save the result into /user/ACCOUNT_NAME/BDA/output folder
average_sort.saveAsTextFile("BDA/output")
```

```
=========================== FINAL OUTPUT ==========================
((u'96000', u'2014-07'), 26.3)
((u'65450', u'1994-07'), 23.65483870967742)
((u'95160', u'1994-07'), 23.505376344086027)
((u'75120', u'1994-07'), 23.26881720430107)
((u'105260', u'1994-07'), 23.143820224719107)
((u'85280', u'1994-07'), 23.108602150537635)
((u'54550', u'1983-08'), 23.0)
((u'54550', u'1975-08'), 22.9625)
((u'96550', u'1994-07'), 22.957894736842114)
((u'96000', u'1994-07'), 22.931182795698923)
((u'106070', u'1994-07'), 22.822580645161295)
((u'173960', u'1972-07'), 22.776666666666667)
((u'54300', u'1994-07'), 22.76021505376344)
((u'85210', u'1994-07'), 22.755913978494615)
((u'65450', u'2006-07'), 22.74086021505376)
```

Alice Velander, alive213
David Björelind, davbj395

4)
Code:



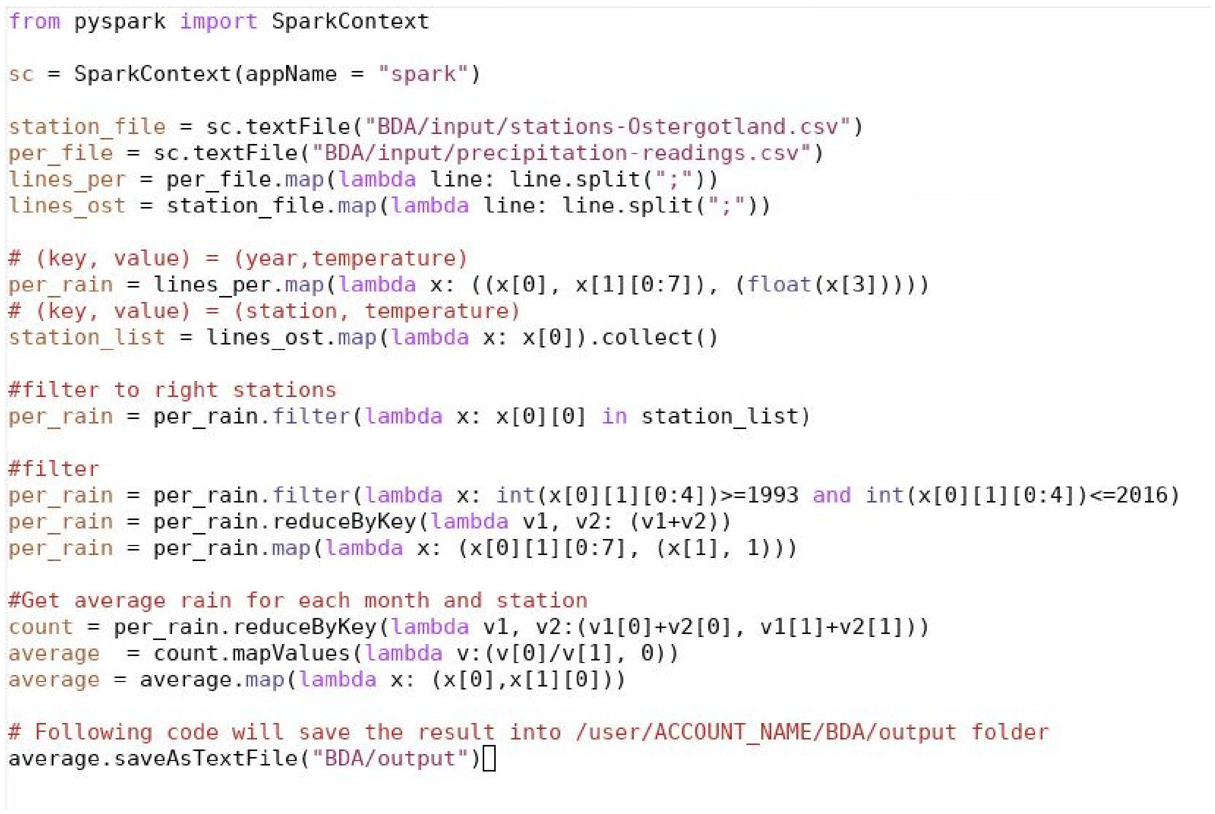
```
from pyspark import SparkContext

sc = SparkContext(appName = "spark")

temperature_file = sc.textFile("BDA/input/temperature-readings.csv")
per_file = sc.textFile("BDA/input/precipitation-readings.csv")
lines_temp = temperature_file.map(lambda line: line.split(";"))
lines_per = per_file.map(lambda line: line.split(";"))

def max(a, b):
    if a>=b:
        return a
    else:
        return b

# (key, value) = (station, temperature)
#stat_temperature = lines_temp.map(lambda x: ((x[0], x[1]), float(x[3])))
stat_temperature = lines_temp.map(lambda x: (x[0], float(x[3])))
# (key, value) = (station, percipitation)
stat_per = lines_per.map(lambda x: ((x[0], x[1]), float(x[3])))
# Getting daily reading for each station (instead of hourly)
stat_per = stat_per.reduceByKey(lambda x, y: x+y)
stat_per = stat_per.map(lambda x: (x[0][0], x[1]))

#get max temp for each station and day
max_temp = stat_temperature.reduceByKey(max)
#filter
some_temperature = max_temp.filter(lambda x: int(x[1]>=25) and int(x[1]<=30))

#max rain for each station and day
max_rain = stat_per.reduceByKey(max)
#filter rain
some_rain = max_rain.filter(lambda x: int(x[1]<=200) and int(x[1]>=100))

#join
data=some_rain.join(some_temperature)

data.saveAsTextFile("BDA/output")
```

========================== FINAL OUTPUT ==========================
==

No output!

Alice Velander, alive213
David Björelind, davbj395

5)
Code:



```
from pyspark import SparkContext

sc = SparkContext(appName = "spark")

station_file = sc.textFile("BDA/input/stations-Ostergotland.csv")
per_file = sc.textFile("BDA/input/precipitation-readings.csv")
lines_per = per_file.map(lambda line: line.split(";"))
lines_ost = station_file.map(lambda line: line.split(";"))

# (key, value) = (year,temperature)
per_rain = lines_per.map(lambda x: ((x[0], x[1][0:7]), (float(x[3]))))
# (key, value) = (station, temperature)
station_list = lines_ost.map(lambda x: x[0]).collect()

#filter to right stations
per_rain = per_rain.filter(lambda x: x[0][0] in station_list)

#filter
per_rain = per_rain.filter(lambda x: int(x[0][1][0:4])>=1993 and int(x[0][1][0:4])<=2016)
per_rain = per_rain.reduceByKey(lambda v1, v2: (v1+v2))
per_rain = per_rain.map(lambda x: (x[0][1][0:7], (x[1], 1)))

#Get average rain for each month and station
count = per_rain.reduceByKey(lambda v1, v2:(v1[0]+v2[0], v1[1]+v2[1]))
average  = count.mapValues(lambda v:(v[0]/v[1], 0))
average = average.map(lambda x: (x[0],x[1][0]))

# Following code will save the result into /user/ACCOUNT_NAME/BDA/output folder
average.saveAsTextFile("BDA/output")
```

```
=========================== FINAL OUTPUT ===========================
(u'1996-11', 67.11666666666665)
(u'2008-10', 59.566666666666684)
(u'2014-05', 58.00000000000001)
(u'2001-11', 26.38333333333334)
(u'2011-05', 37.85)
(u'2010-09', 43.08333333333335)
(u'2010-02', 52.75000000000005)
(u'2013-08', 54.075)
(u'2007-02', 33.06666666666668)
(u'2002-06', 98.78333333333333)
(u'2013-05', 47.92500000000001)
(u'1998-11', 28.96666666666668)
(u'2002-03', 26.93333333333334)
(u'2013-02', 25.52500000000002)
(u'2000-01', 18.61666666666667)
```