B7

1. True or false problems with wrong-answer penalties:
    a. False
    b. True
    c. True
    d. False
    e. False
    f. True
    g. True
    h. False
2. What is the output of the following pyspark code: “Pythonprogrammingisawesome!”
3. What is the output of the following pyspark code:

    unionDF.show(truncate=False)

employee_name	department	state	salary	age	bonus
James	Sales	NY	90000	34	10000
Michael	Sales	NY	86000	56	20000
Robert	Sales	CA	81000	30	23000
Maria	Finance	CA	90000	24	23000
James	Sales	NY	90000	34	10000
Maria	Finance	CA	90000	24	23000
Jen	Finance	NY	79000	53	15000
Jeff	Marketing	CA	80000	25	18000
Kumar	Marketing	NY	91000	50	21000

    disDF.show(truncate=False)

employee_name	department	state	salary	age	bonus
James	Sales	NY	90000	34	10000

Michael	Sales	NY	86000	56	20000
Robert	Sales	CA	81000	30	23000
Maria	Finance	CA	90000	24	23000
Jen	Finance	NY	79000	53	15000
Jeff	Marketing	CA	80000	25	18000
Kumar	Marketing	NY	91000	50	21000

unionAllDF.show(truncate=False)

employee_name	department	state	salary	age	bonus
James	Sales	NY	90000	34	10000
Michael	Sales	NY	86000	56	20000
Robert	Sales	CA	81000	30	23000
Maria	Finance	CA	90000	24	23000
James	Sales	NY	90000	34	10000
Maria	Finance	CA	90000	24	23000
Jen	Finance	NY	79000	53	15000
Jeff	Marketing	CA	80000	25	18000
Kumar	Marketing	NY	91000	50	21000

4. Given a test file “input.dat” with empty, please write a PySpark code to output number of empty lines in the file

```
from pyspark import SparkContext
spark = SparkContext()
df = spark.textFile("./input.dat")
df = df.filter(lambda line: line.strip() == "")

print(df.count())
```

5. Please write a MapReduce pseudocode to solve the grouping and aggregation problem.

Let R(A, B, C) be a relation to which we apply the operator gA,q(B)(R).

```
group_by_col = "A"
aggregation_function = "sum"

grouped_data = data.groupBy(group_by_col)


aggregated_data = grouped_data.agg(
   col("B").alias(aggregation_function)(aggregation_function)  )
```

6. Suppose there is a repository of 2^23 documents, and word w appears in 512 of them. In a particular document d, the maximum number of occurrences of a word is 70. Approximately what is the TF.IDF score for w if that word appears (a) once (b) 5 times?

   a) 14
   b) 70

7. Using the matrix-vector multiplication, applied to the matrix and vector:
   1 2 3 4 1
   5 6 7 8 2
   9 10 11 12 3
   13 14 15 16 4
   apply the Map function to this matrix and vector (assume the vector can load into memory fully). Then, identify in the list below, one of the key-value pairs that are output of Map.
   a) (3, 45)
   b) (1, 13)
   c) (2, 70)
   d) (4, 28)

   The answer is C (2, 70)

8. Which of the following will cause a core dump? What is the output if it can execute without core dump?

   The answer is C.

   For the code to execute we have to return an iterable from the function in flatMap

   i.e we can change it to
   r = data.flatMap(lambda x: [x])

9. The following program changes CourseName to uppercase, reduces price by 1000 for each course, and then applies a different discount to reach the course. Please replace

the pass with correct code to make it working. [Hint: use df.withColumn()]

```
from pyspark.sql.functions import upper, col, lit
from pyspark.sql import SparkSession
spark = SparkSession.builder \
   .appName('SparkByExamples.com') \
   .getOrCreate()
simpleData = (("Java", 4000, 5),
             ("Python", 4600, 10),  ("Scala", 4100, 15),  ("PHP", 3000, 20),
             )
columns = ["CourseName", "fee", "discount"]
df = spark.createDataFrame(data=simpleData, schema=columns)


def to_upper_str_columns(df):
   return df.withColumn("CourseName",
col("CourseName").cast("string")).withColumn("CourseName",
upper(col("CourseName")))


def reduce_price(df, reduceBy):
   return df.withColumn("fee", col("fee") - reduceBy)


def apply_discount(df):

   return df.withColumn("fee", col("fee") - 1000)


df2 = df.transform(to_upper_str_columns).transform(reduce_price, 1000) \
   .transform(apply_discount)

df2.show()
```

10. Please replace pass with correct code to make it working

```
from pyspark import SparkConf, SparkContext
from pyspark.sql import SQLContext
sc = SparkContext()
sqlContext = SQLContext(sc)
```

```
company_df = sqlContext.read.format('com.databricks.spark.csv') \
.options(header='true',inferschema='true') \
.load('Fortune5002017.csv') company_df.printSchema()

from pyspark.ml.feature import VectorAssembler

vectorAssembler = VectorAssembler(
   inputCols=["Employees"], outputCol="features")
tcompany_df = vectorAssembler.transform(company_df)


tcompany_df = tcompany_df.select(["features", "Employees"])
tcompany_df.show(3)

splits = tcompany_df.randomSplit([0.7, 0.3])
train_df = splits[0]
test_df = splits[1]



lr = LinearRegression(featuresCol="features", labelCol="Employees",
                      maxIter=10, regParam=0.3, elasticNetParam=0.8)

lr_model = lr.fit(train_df)

print("Coefficients: " + str(lr_model.coefficients))
print("Intercept: " + str(lr_model.intercept))


predictions = lr_model.transform(test_df)
```