Distributed architectures for big data processing and analytics

Answer to the following questions. There is only one right answer for each question.

1. (2 points) Consider the following Spark Streaming applications.

(Application A)

```
from pyspark.streaming import StreamingContext

# Create a Spark Streaming Context object
ssc = StreamingContext(sc, 10)

# Create a (Receiver) DStream that will connect to localhost:9999
# Define windows
inputWindowDStream = ssc.socketTextStream("localhost", 9999)\
.window(20, 10)\
.map(lambda value: int(value))

# Sum values
sumWindowDStream = inputWindowDStream.reduce(lambda v1,v2: v1+v2)

#Apply a filter
resDStream = sumWindowDStream.filter(lambda value : value>5)

# Print the result
resDStream.pprint()

ssc.start()
ssc.awaitTerminationOrTimeout(360)
ssc.stop(stopSparkContext=False)
```

(Application B)

```
from pyspark.streaming import StreamingContext

# Create a Spark Streaming Context object
ssc = StreamingContext(sc, 10)

# Create a (Receiver) DStream that will connect to localhost:9999
inputDStream = ssc.socketTextStream("localhost", 9999)\
.map(lambda value: int(value))

# Sum values
sumDStream = inputDStream.reduce(lambda v1,v2: v1+v2)

# Define windows
sumWindowDStream = sumDStream.window(20, 10)

#Apply a filter
resDStream = sumWindowDStream.filter(lambda value : value>5)

# Print the result
resDStream.pprint()
```

```
ssc.start()
ssc.awaitTerminationOrTimeout(360)
ssc.stop(stopSparkContext=False)
```

(Application C)

```
from pyspark.streaming import StreamingContext

# Create a Spark Streaming Context object
ssc = StreamingContext(sc, 10)

# Create a (Receiver) DStream that will connect to localhost:9999
inputDStream = ssc.socketTextStream("localhost", 9999)\
.map(lambda value: int(value))

# Define windows
inputWindowDStream = inputDStream.window(20, 10)

# Sum values
sumWindowDStream = inputWindowDStream.reduce(lambda v1,v2: v1+v2)

#Apply a filter
resDStream = sumWindowDStream.filter(lambda value : value>5)

# Print the result
resDStream.pprint()

ssc.start()
ssc.awaitTerminationOrTimeout(360)
ssc.stop(stopSparkContext=False)
```

Which one of the following statements is true?

a) Applications A, B, And C are equivalent in terms of returned result, i.e., given the same input they return the same result.

b) Applications A and B are equivalent in terms of returned result, i.e., given the same input they return the same result, while C is not equivalent to the other two applications.

c) Applications A and C are equivalent in terms of returned result, i.e., given the same input they return the same result, while B is not equivalent to the other two applications.

d) Applications B and C are equivalent in terms of returned result, i.e., given the same input they return the same result, while A is not equivalent to the other two applications.

2. (2 points) Consider the following Spark Streaming applications.

(Application A)

```
from pyspark.streaming import StreamingContext

# Create a Spark Streaming Context object
ssc = StreamingContext(sc, 10)

# Create a (Receiver) DStream that will connect to localhost:9999
# Define windows
inputWindowDStream = ssc.socketTextStream("localhost", 9999)\
.window(20, 10)\
.map(lambda value: int(value))

# Sum values
sumWindowDStream = inputWindowDStream.reduce(lambda v1,v2: v1+v2)

#Apply a filter
resDStream = sumWindowDStream.filter(lambda value : value>5)

# Print the result
resDStream.pprint()

ssc.start()
ssc.awaitTerminationOrTimeout(360)
ssc.stop(stopSparkContext=False)
```

(Application B)

```
from pyspark.streaming import StreamingContext

# Create a Spark Streaming Context object
ssc = StreamingContext(sc, 10)

# Create a (Receiver) DStream that will connect to localhost:9999
inputDStream = ssc.socketTextStream("localhost", 9999)\
.map(lambda value: int(value))

# Sum values
sumDStream = inputDStream.reduce(lambda v1,v2: v1+v2)

# Define windows
sumWindowDStream = sumDStream.window(20, 10)

#Apply a filter
resDStream = sumWindowDStream.filter(lambda value : value>5)

# Print the result
resDStream.pprint()

ssc.start()
ssc.awaitTerminationOrTimeout(360)
ssc.stop(stopSparkContext=False)
```

(Application C)

```
from pyspark.streaming import StreamingContext

# Create a Spark Streaming Context object
ssc = StreamingContext(sc, 10)

# Create a (Receiver) DStream that will connect to localhost:9999
inputDStream = ssc.socketTextStream("localhost", 9999)\
.map(lambda value: int(value))

# Sum values
sumDStream = inputDStream.reduce(lambda v1,v2: v1+v2)

#Apply a filter
sumFilterDStream = sumDStream.filter(lambda value : value>5)

# Define windows
resDStream = sumFilterDStream.window(20, 10)

# Print the result
resDStream.pprint()

ssc.start()
ssc.awaitTerminationOrTimeout(360)
ssc.stop(stopSparkContext=False)
```

Which one of the following statements is true?

a) Applications A, B, And C are equivalent in terms of returned result, i.e., given the same input they return the same result.

b) Applications A and B are equivalent in terms of returned result, i.e., given the same input they return the same result, while C is not equivalent to the other two applications.

c) Applications A and C are equivalent in terms of returned result, i.e., given the same input they return the same result, while B is not equivalent to the other two applications.

d) Applications B and C are equivalent in terms of returned result, i.e., given the same input they return the same result, while A is not equivalent to the other two applications.

3. Which one of the following sentences about SparkStreaming is True?

a) Micro-batch execution offers lower latency than continuous execution

b) Structured Streaming API works with DStream objects

c) Two StreamingContext can be run in parallel on the same application

d) SparkStreaming provide Checkpoints to recover from failures

4. (2 points) Consider the following piece of Spark application:

```
myrdd=sc.parallelize(['I','am','an','example'])
myacc=sc.accumulator(0)

def strFun(line):
    l=len(line)
    if l<3:
        myacc.add(1)
    return l

v=myrdd.map(lambda l:strFun(l)).reduce(lambda l1,l2:l1+l2)

print(myacc.value,v)
```

What will be the output of the application?

a) 3 4
b) 0 'example'
c) 0 'Iamanexample'
d) 3 12

5. (2 points) Which one of the following statement about the preprocessing of data in Mllib is true?

a) A "vectorAssembler" estimator always merges all the columns of a Dataframe into a single vector

b) A "vectorAssembler" estimator can be applied to columns of type double

c) "indexToString" is useful to apply in preprocessing before training the a prediction model

d) "standardScaler" can be applied to strings

6. (2 points) Consider the following piece of Spark application.

```
from graphframes import GraphFrame

v = spark.read.load(inputPath1, format='csv',inferSchema=True).toDF("id","attribute")

e = spark.read.load(inputPath2, format='csv',inferSchema=True).toDF("src","dst","attribute")
```

```
g = GraphFrame(v, e)
n=g.vertices.count()
e2 = e.filter("attribute = 'typeA'")
g2 = GraphFrame(g.vertices, e2).dropIsolatedVertices()
```

Which one of the following statements is **false**?

a) The variable “n” is an integer number

b) “g.vertices” is a dataframe

c) “g2” is a subgraph of “g”

d) “e2” is a graphframe