Spark streaming - Multiple choice questions - Examples

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)

```
import ..;
public class SparkDriver {
	public static void main(String[] args) throws InterruptedException {
		// Create a configuration object and set the name of the application
		SparkConf conf = new SparkConf().setAppName("Spark Streaming A");

		// Create a Spark Streaming Context object
		JavaStreamingContext jssc = new JavaStreamingContext(conf,
							Durations.seconds(10));

		// Define a DStream associated with the TPC socket localhost:9999
		// Apply window and map the input strings to integers
		JavaDStream<Integer> inputWindowDStream = jssc
				.socketTextStream("localhost", 9999)
				.window(Durations.seconds(20), Durations.seconds(10))
				.map(value -> Integer.valueOf(value));

		// Sum values
		JavaDStream<Integer> sumWindowDStream = inputWindowDStream
				.reduce((v1,v2) -> v1 + v2);

		// Apply a filter
		JavaDStream<Integer> resDStream = sumWindowDStream
				.filter(value -> value>5);

		// Print the result on the standard output
		resDStream.print();


		// Start the computation
		jssc.start();
		jssc.awaitTerminationOrTimeout(360);
		jssc.close();
	}
}
```

(Application B)

```
import ..;
public class SparkDriver {
	public static void main(String[] args) throws InterruptedException {
		// Create a configuration object and set the name of the application
		SparkConf conf = new SparkConf().setAppName("Spark Streaming B");

		// Create a Spark Streaming Context object
		JavaStreamingContext jssc = new JavaStreamingContext(conf,
							Durations.seconds(10));

		// Define a DStream associated with the TPC socket localhost:9999
		// Map the input strings to integers
		JavaDStream<Integer> inputDStream = jssc.socketTextStream("localhost", 9999)
				.map(value -> Integer.valueOf(value));

		// Sum values
		JavaDStream<Integer> sumDStream = inputDStream.reduce((v1, v2) -> v1 + v2);

		// Define windows
		JavaDStream<Integer> sumWindowDStream = sumDStream
				.window(Durations.seconds(20), Durations.seconds(10));

		// Apply a filter
		JavaDStream<Integer> resDStream = sumWindowDStream
				.filter(value -> value>5);

		// Print the result on the standard output
		resDStream.print();

		// Start the computation
		jssc.start();
		jssc.awaitTerminationOrTimeout(360);
		jssc.close();
	}
}
```

(Application C)

```
import ..;
public class SparkDriver {
        public static void main(String[] args) throws InterruptedException {
                // Create a configuration object and set the name of the application
                SparkConf conf = new SparkConf().setAppName("Spark Streaming C");

                // Create a Spark Streaming Context object
                JavaStreamingContext jssc = new JavaStreamingContext(conf,
                                                        Durations.seconds(10));

                // Define a DStream associated with the TPC socket localhost:9999
                JavaDStream<Integer> inputDStream = jssc.socketTextStream("localhost", 9999)
                                .map(value -> Integer.valueOf(value));

                // Define windows
                JavaDStream<Integer> inputWindowDStream = inputDStream
                                .window(Durations.seconds(20), Durations.seconds(10));

                // Sum values
                JavaDStream<Integer> sumWindowDStream = inputWindowDStream
                                .reduce((v1, v2) -> v1 + v2);

                // Apply a filter
                JavaDStream<Integer> resDStream = sumWindowDStream
                                .filter(value -> value>5);

                // Print the result on the standard output
                resDStream.print();

                // Start the computation
                jssc.start();
                jssc.awaitTerminationOrTimeout(360);
                jssc.close();
        }
}
```

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)

```
import ..;
public class SparkDriver {
        public static void main(String[] args) throws InterruptedException {
                // Create a configuration object and set the name of the application
                SparkConf conf = new SparkConf().setAppName("Spark Streaming A");

                // Create a Spark Streaming Context object
                JavaStreamingContext jssc = new JavaStreamingContext(conf,
                                                        Durations.seconds(10));

                // Define a DStream associated with the TPC socket localhost:9999
                // Apply window and map the input strings to integers
                JavaDStream<Integer> inputWindowDStream = jssc
                                .socketTextStream("localhost", 9999)
                                .window(Durations.seconds(20), Durations.seconds(10))
                                .map(value -> Integer.valueOf(value));

                // Sum values
                JavaDStream<Integer> sumWindowDStream = inputWindowDStream
                                .reduce((v1,v2) -> v1 + v2);

                // Apply a filter
                JavaDStream<Integer> resDStream = sumWindowDStream
                                .filter(value -> value>5);

                // Print the result on the standard output
                resDStream.print();


                // Start the computation
                jssc.start();
                jssc.awaitTerminationOrTimeout(360);
                jssc.close();
        }
}
```

(Application B)

```
import ..;
public class SparkDriver {
	public static void main(String[] args) throws InterruptedException {
		// Create a configuration object and set the name of the application
		SparkConf conf = new SparkConf().setAppName("Spark Streaming B");

		// Create a Spark Streaming Context object
		JavaStreamingContext jssc = new JavaStreamingContext(conf,
							Durations.seconds(10));

		// Define a DStream associated with the TPC socket localhost:9999
		// Map the input strings to integers
		JavaDStream<Integer> inputDStream = jssc.socketTextStream("localhost", 9999)
				.map(value -> Integer.valueOf(value));

		// Sum values
		JavaDStream<Integer> sumDStream = inputDStream.reduce((v1, v2) -> v1 + v2);

		// Define windows
		JavaDStream<Integer> sumWindowDStream = sumDStream
				.window(Durations.seconds(20), Durations.seconds(10));

		// Apply a filter
		JavaDStream<Integer> resDStream = sumWindowDStream
				.filter(value -> value>5);

		// Print the result on the standard output
		resDStream.print();

		// Start the computation
		jssc.start();
		jssc.awaitTerminationOrTimeout(360);
		jssc.close();
	}
}
```

(Application C)

```
import ..;
public class SparkDriver {
        public static void main(String[] args) throws InterruptedException {
                // Create a configuration object and set the name of the application
                SparkConf conf = new SparkConf().setAppName("Spark Streaming C");

                // Create a Spark Streaming Context object
                JavaStreamingContext jssc = new JavaStreamingContext(conf,
                                                        Durations.seconds(10));

                // Define a DStream associated with the TPC socket localhost:9999
                // Map the input strings to integers
                JavaDStream<Integer> inputDStream = jssc.socketTextStream("localhost", 9999)
                                .map(value -> Integer.valueOf(value));

                // Sum values
                JavaDStream<Integer> sumDStream = inputDStream.reduce((v1, v2) -> v1 + v2);

                //Apply a filter
                JavaDStream<Integer> sumFilterDStream = sumDStream.filter(value -> value>5)

                // Define windows
                JavaDStream<Integer> resDStream = sumFilterDStream
                                .window(Durations.seconds(20), Durations.seconds(10));


                // Print the result on the standard output
                resDStream.print();

                // Start the computation
                jssc.start();
                jssc.awaitTerminationOrTimeout(360);
                jssc.close();
        }
}
```

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

```
import ..;
public class SparkDriver {
        public static void main(String[] args) throws InterruptedException {
                // Create a Spark Streaming Context object
                JavaStreamingContext jssc = new JavaStreamingContext(conf,
                                                        Durations.seconds(10));

                // Define a DStream associated with the TPC socket localhost:9999
                // Map the input strings to integers
                JavaDStream<Integer> inputDStream = jssc.socketTextStream("localhost", 9999)
                                .map(value -> Integer.valueOf(value));

                // Sum values
                JavaDStream<Integer> sumDStream = inputDStream.reduce((v1, v2) -> v1 + v2);

                // Define windows
                JavaDStream<Integer> resDStream = sumDStream
                                .window(Durations.seconds(20), Durations.seconds(10));


                // Print the result on the standard output
                resDStream.print();

                // Start the computation
                jssc.start();
                jssc.awaitTerminationOrTimeout(360);
                jssc.close();
        }
}
```

Consider the following input data
Time: 1s -> “2“
Time: 3s -> “2”
Time: 5s -> “1”
Time: 12s -> “4”
Time: 14s -> “2”

Which one of the following statements is true?

a) The application, after 20 seconds, prints on the standard output the value 11.

b) The application, after 20 seconds, prints on the standard output the values 5 and 6.

c) The application, after 20 seconds, prints on the standard output the value 6.

d) The application, after 20 seconds, prints on the standard output the value 5.