Strings Test 2

**Multiple Choice**

1. Consider the following code segment.

String str = "dog";

for(int i = 0; i < 4; i++)

{

str = str + "x";

}

System.out.println(str);

What is printed as a result of executing this code segment?

1. dog
2. xxx
3. dogxxx
4. dogxxxx
5. dogxdogxxdogxxxdogxxxx
6. Consider the following code segment.

String str = "football";

System.out.println(str.substring(0,1) + str.substring(5));

What is printed as a result of executing this code segment?

1. foot
2. ball
3. foball
4. fall
5. foall
6. Consider the following code segment.

String str = "horse";

int x = str.indexOf("r");

System.out.print(str.substring(x));

What is printed as a result of executing this code segment?

1. r
2. horse
3. hors
4. orse
5. rse
6. Consider the following code segment.

String str = "friendly reminder";

int index = str.indexOf("r");

System.out.println(index);

What is printed as a result of executing this code segment?

1. -1
2. 1
3. 8
4. 15
5. 0
6. Consider the following code segment.

String str = "I love computer programming";

int index = str.indexOf("r");

int index2 = str.indexOf("r", index + 1);

What is the value of index2 after the code has executed?

1. 14
2. 15
3. 16
4. 17
5. 18

**Free Response**

1. This question involves reasoning about strings that represent social security numbers. You will implement two related methods that appear in the same class. The first method takes a single string parameter representing a social security number and returns true if the social security number contains the correct number of dashes and the dashes are in the correct position. The second method takes a single parameter representing a social security number and returns true if the number is the correct length and the dashes are valid.
2. Write the method **validateDashes**, which takes a given social security number and determines if is in the correct format. A number is in the correct format if there

* are two and only two dashes
* the dashes are located in the correct position (XXX-XX-XXXX)

The following table shows some examples of valid and invalid dash placement.

|  |  |
| --- | --- |
| **Social Security Number** | **Dashes** |
| 462-58-6985 | valid |
| 462586985 | Invalid, must have two dashes |
| 462-58-69-85 | Invalid, must have two dashes |
| 46-258-6985 | Invalid, dashes are not in correct positions |

Complete method **validateDashes** below.

/\* This method returns true if the given social security number

\* has the correct number of dashes and if they are

\* in the correct position; otherwise it returns false.

\* @param ssn a string representing a social security

\* @return true or false

\*/

private static boolean **validateDashes**(String ssn)

1. Write the method **validate**, which takes a given social security number and determines if it is the correct length and if it has the correct number of dashes and they are in the correct position. A valid social security has the following format: XXX-XX-XXXX.

Assume that **validate** is in the same class as **validateDashes** and works as specified, regardless of what you wrote in part (a).

Complete method **validate** below.

/\* This method returns true if the given social security number

\* has the correct format and length.

\* @param ssn a string representing a phone number

\* @return true or false

\*/

public static boolean validate(String ssn)