Strings Test 1 Key

Multiple Choice

1. Consider the following code segment.

```
String str = "abc";
for(int i = 0; i < 3; i++)
{
   str += "x";
}
System.out.println(str);
```

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

A. xxx
B. abc
C. abcx
D. abcxxx
E. abcxabcxxabcxxx

2. Consider the following code segment.

```
String str = "cheerleader";
System.out.println(str.substring(4,5) + str.substring(6));
```

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

A. `erl`
B. leader
C. reader
D. rle
E. re

3. Consider the following code segment.

```
String str = "abcde";
for(int i = 0; i < str.length(); i++)
{
   System.out.print(str.substring(i, i+1));
}
System.out.println();
```

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

A. abcde
B. aaaaa
C. abcd
D. bcde
E. no output because of a StringIndexOutOfBounds exception.

4. Consider the following code segment.

```
String str = "manifest destiny";
int index = str.indexOf("if");
System.out.println(index);
```

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

A. 0
B. 3
C. -1
D. 2
E. 16

5. Consider the following code segment.

```
String str = "I love computer programming";
int num = 0;
int index = 0;
while(index >=0)
{
   index = str.indexOf("o", index);
   if(index >= 0)
   {
      num += index;
      index++;
   }
}
System.out.println(num);
```

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

A. 0
B. 3
C. 8
D. 27
E. 29

Free Response

1. This question involves reasoning about strings that represent phone numbers. You will implement two related methods that appear in the same class. The first method takes a single string parameter representing a phone number and returns true if the phone number contains the correct number of dashes and the dashes are in the correct position. The second method takes a single parameter representing a phone number and returns true if the number is the correct length and the dashes are valid.

   (a) Write the method **validateDashes**, which takes a given phone 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-XXX-XXXX)

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

Phone Number	Dashes
940-562-5487	valid
9405625487	Invalid, must have two dashes
940-5625-48-7	Invalid, must have two dashes
940-56-2487	Invalid, dashes are not in correct positions

Complete method **validateDashes** below.

Version 1

```
/* This method returns true if the given phone number
 * has the correct number of dashes and if they are
 * in the correct position; otherwise it returns false.
 * @param phonenumber a string representing a phone number
 * @return true or false
 */
private static boolean validateDashes(String phoneNumber)
{
   int dash1 = phoneNumber.indexOf("-");
   int dash2 = phoneNumber.indexOf("-", 4);
   int dash3 = phoneNumber.indexOf("-", 8);

   if(dash1 == 3 && dash2 == 7 && dash3 == -1)
       return true;
   else
       return false;
}
```

Version 2

```
private static boolean validateDashes(String phoneNumber)
   {
       String dash1 = phoneNumber.substring(3,4);
       String dash2 = phoneNumber.substring(7,8);
       int numDashes = 0;

       for(int i = 0; i < phoneNumber.length(); i++)
       {
          if(phoneNumber.substring(i, i+1).equals("-"))
          {
               numDashes++;
          }
       }

       if(dash1.equals("-") && dash2.equals("-") && numDashes == 2)
           return true;
       else
           return false;
   }
```

(b) Write the method **validate**, which takes a given phone 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 phone number has the following format: XXX-XXX-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 phone number
 * has the correct format and length.
 * @param phonenumber a string representing a phone number
 * @return true or false
 */
public static boolean validate(String phoneNumber)
{
   return validateDashes(phoneNumber) &&
          phoneNumber.length() == 12;
}
```