This question involves the use of *check digits,* which can be used to help detect if an error has occurred when a number is entered or transmitted electronically. An algorithm for computing a check digit, based on the digits of a number, is provided in part (a).

The `CheckDigit` class is shown below. You will write two methods of the `CheckDigit` class.

```
public class CheckDigit
{
    /** Returns the check digit for num, as described in part (a).
     *  Precondition: The number of digits in num is between one and
     *  six, inclusive.
     *            num >= 0
     */
     public static int getCheck(int num)
     {
         /* to be implemented in part (a) */
     }

    /** Returns true if numWithCheckDigit is valid, or false
     *  otherwise, as described in part (b).
     *  Precondition: The number of digits in numWithCheckDigit
     *  is between two and seven, inclusive.
     *              numWithCheckDigit >= 0
     */
    public static boolean isValid(int numWithCheckDigit)
    {
      /* to be implemented in part (b) */
    }

    /** Returns the number of digits in num. */
    public static int getNumberOfDigits(int num)
    {
      /* implementation not shown */
    }

    /** Returns the nthdigit of num.
     *  Precondition: n >= 1 and n <= the number of digits in num
     */
    public static int getDigit(int num, int n)
    {
      /* implementation not shown */
    }

    // There may be instance variables, constructors, and methods not shown.
}
```

(a) Complete the `getCheck` method, which computes the check digit for a number according to the following rules.

- Multiply the first digit by 7, the second digit (if one exists) by 6, the third digit (if one exists) by 5, and so on. The length of the method's `int` parameter is at most six; therefore, the last digit of a six-digit number will be multiplied by 2.
- Add the products calculated in the previous step.
- Extract the check digit, which is the rightmost digit of the sum calculated in the previous step.

The following are examples of the check-digit calculation.

Example 1, where num has the value 283415

- The sum to calculate is $(2 \times 7) + (8 \times 6) + (3 \times 5) + (4 \times 4) + (1 \times 3) + (5 \times 2) = 14 + 48 + 15 + 16 + 3 + 10 = 106$.
- The check digit is the rightmost digit of 106, or 6, and `getCheck` returns the integer value `6`.

Example 2, where num has the value 2183

- The sum to calculate is $(2 \times 7) + (1 \times 6) + (8 \times 5) + (3 \times 4) = 14 + 6 + 40 + 12 = 72$.
- The check digit is the rightmost digit of 72, or 2, and `getCheck` returns the integer value `2`.

Two helper methods, `getNumberOfDigits` and `getDigit`, have been provided.

- `getNumberOfDigits` returns the number of digits in its `int` parameter.
- `getDigit` returns the `n`th digit of its `int` parameter.

The following are examples of the use of `getNumberOfDigits` and `getDigit`.

Method Call	Return Value	Explanation
`getNumberOfDigits(283415)`	6	The number `283415` has 6 digits.
`getDigit(283415, 1)`	2	The first digit of `283415` is `2`.
`getDigit(283415, 5)`	1	The fifth digit of `283415` is `1`.

Complete the `getCheck` method below. You must use `getNumberOfDigits` and `getDigit` appropriately to receive full credit.

```
/** Returns the check digit for num, as described in part (a).
 *  Precondition: The number of digits in num is between one and six,
 *  inclusive.
 *                num >= 0
 */
public static int getCheck(int num)
```

(b) Write the `isValid` method. The method returns `true` if its parameter `numWithCheckDigit`, which represents a number containing a check digit, is valid, and `false` otherwise. The check digit is always the rightmost digit of `numWithCheckDigit`.

The following table shows some examples of the use of `isValid`.

Method Call	Return Value	Explanation
`getCheck(159)`	2	The check digit for 159 is 2.
`isValid(1592)`	`true`	The number 1592 is a valid combination of a number (159) and its check digit (2).
`isValid(1593)`	`false`	The number 1593 is not a valid combination of a number (159) and its check digit (3) because 2 is the check digit for 159.

Complete method `isValid` below. Assume that `getCheck` works as specified, regardless of what you wrote in part (a). You must use `getCheck` appropriately to receive full credit.

```
/** Returns true if numWithCheckDigit is valid, or false
 *  otherwise, as described in part (b).
 *  Precondition: The number of digits in numWithCheckDigit is
 *  between two and seven, inclusive.
 *                  numWithCheckDigit >= 0
 */
public static boolean isValid(int numWithCheckDigit)
```