Q1

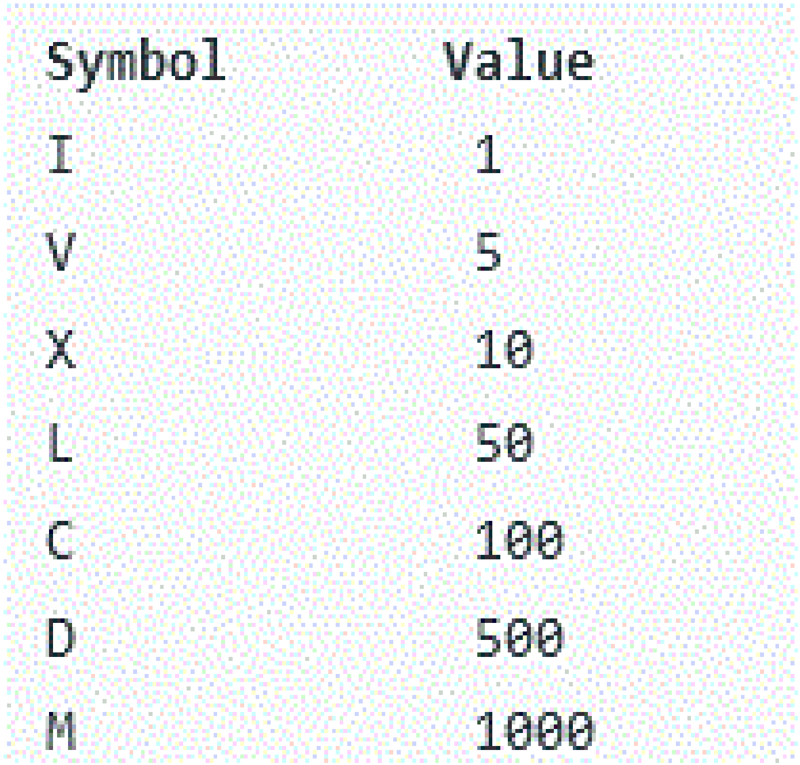
- *(Smallest of Three Numbers)* Write a program that defines and uses macro MINIMUM3 to determine the smallest of three numeric values.

Q2

- *(Printing an Array)* Write a program that defines and uses macro PRINTARRAY to print an array of integers. The macro should receive the array and the number of elements in the array as arguments.

Q3

- Roman numerals are represented by seven different symbols: I, V, X, L, C, D and M.



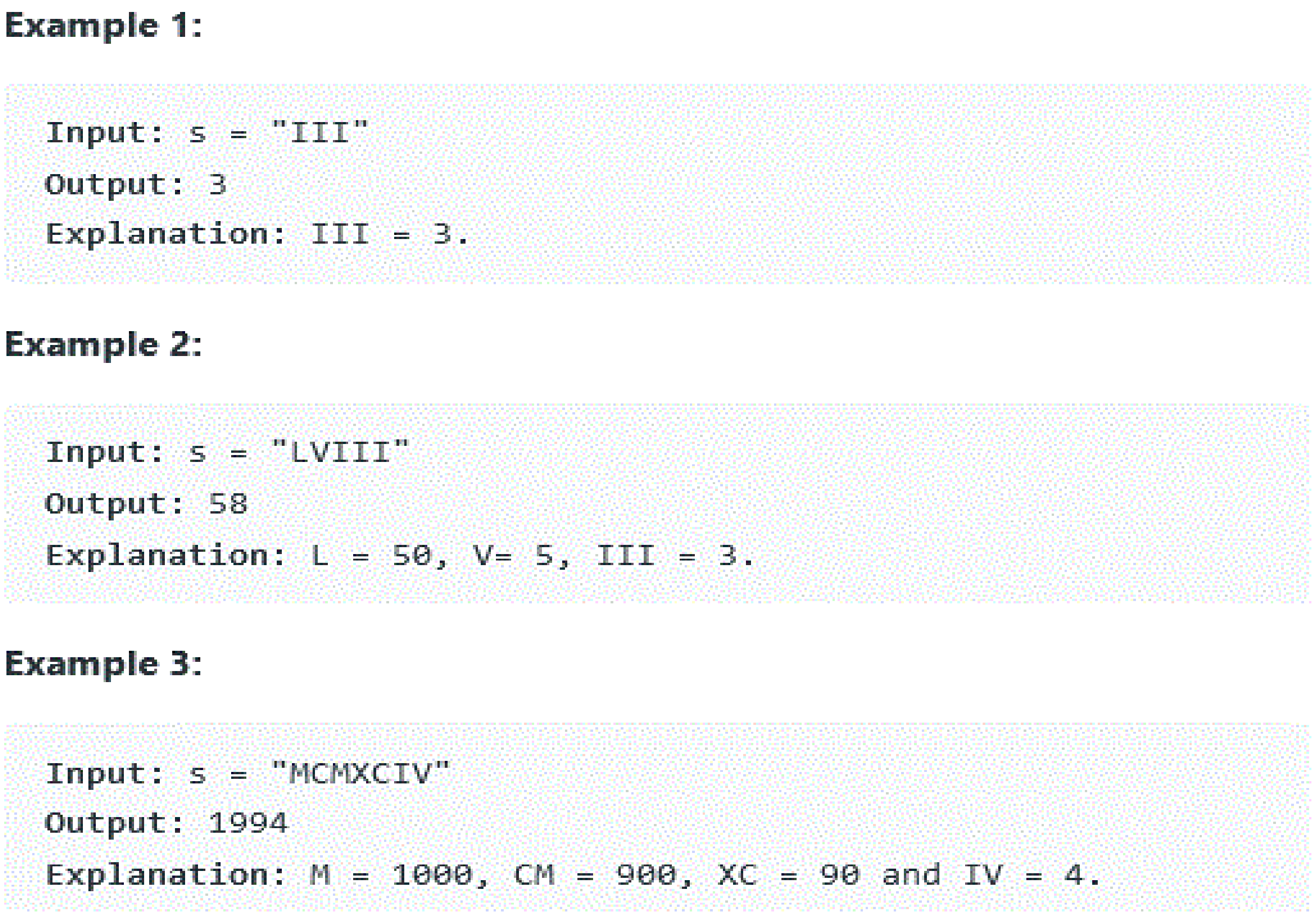
Symbol	Value
I	1
V	5
X	10
L	50
C	100
D	500
M	1000

Q3

- For example, 2 is written as II in Roman numeral, just two one's added together. 12 is written as XII, which is simply X + II. The number 27 is written as XXVII, which is XX + V + II.
- Roman numerals are usually written largest to smallest from left to right. However, the numeral for four is not IIII. Instead, the number four is written as IV. Because the one is before the five we subtract it making four. The same principle applies to the number nine, which is written as IX. There are six instances where subtraction is used:
- I can be placed before V (5) and X (10) to make 4 and 9.
- X can be placed before L (50) and C (100) to make 40 and 90.
- C can be placed before D (500) and M (1000) to make 400 and 900.
- Given a roman numeral, convert it to an integer.

Q3

int romanToInt (char * s);



Example 1:

```
Input: s = "III"
Output: 3
Explanation: III = 3.
```

Example 2:

```
Input: s = "LVIII"
Output: 58
Explanation: L = 50, V= 5, III = 3.
```

Example 3:

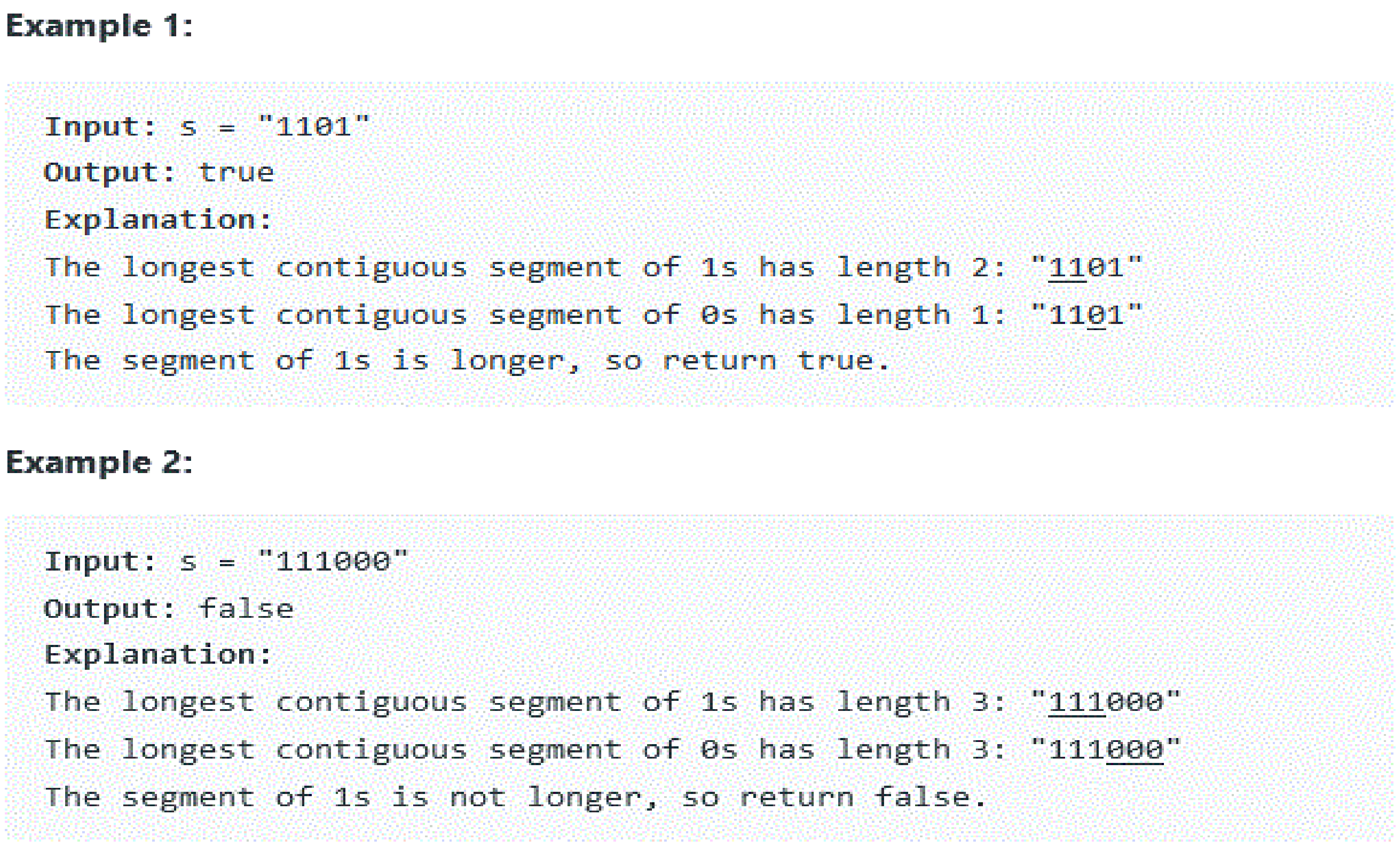
```
Input: s = "MCMXCIV"
Output: 1994
Explanation: M = 1000, CM = 900, XC = 90 and IV = 4.
```

Q4

- Given a binary string s, return true *if the **longest** contiguous segment of* 1'*s is **strictly longer** than the **longest** contiguous segment of* 0'*s in* s, or return false *otherwise*.
- For example, in s = "$\underline{11}01\underline{000}10$" the longest continuous segment of 1s has length 2, and the longest continuous segment of 0s has length 3.
- Note that if there are no 0's, then the longest continuous segment of 0's is considered to have a length 0. The same applies if there is no 1's.

Q4

bool checkZeroOnes(char * s)



Example 1:

```
Input: s = "1101"
Output: true
Explanation:
The longest contiguous segment of 1s has length 2: "1101"
The longest contiguous segment of 0s has length 1: "1101"
The segment of 1s is longer, so return true.
```

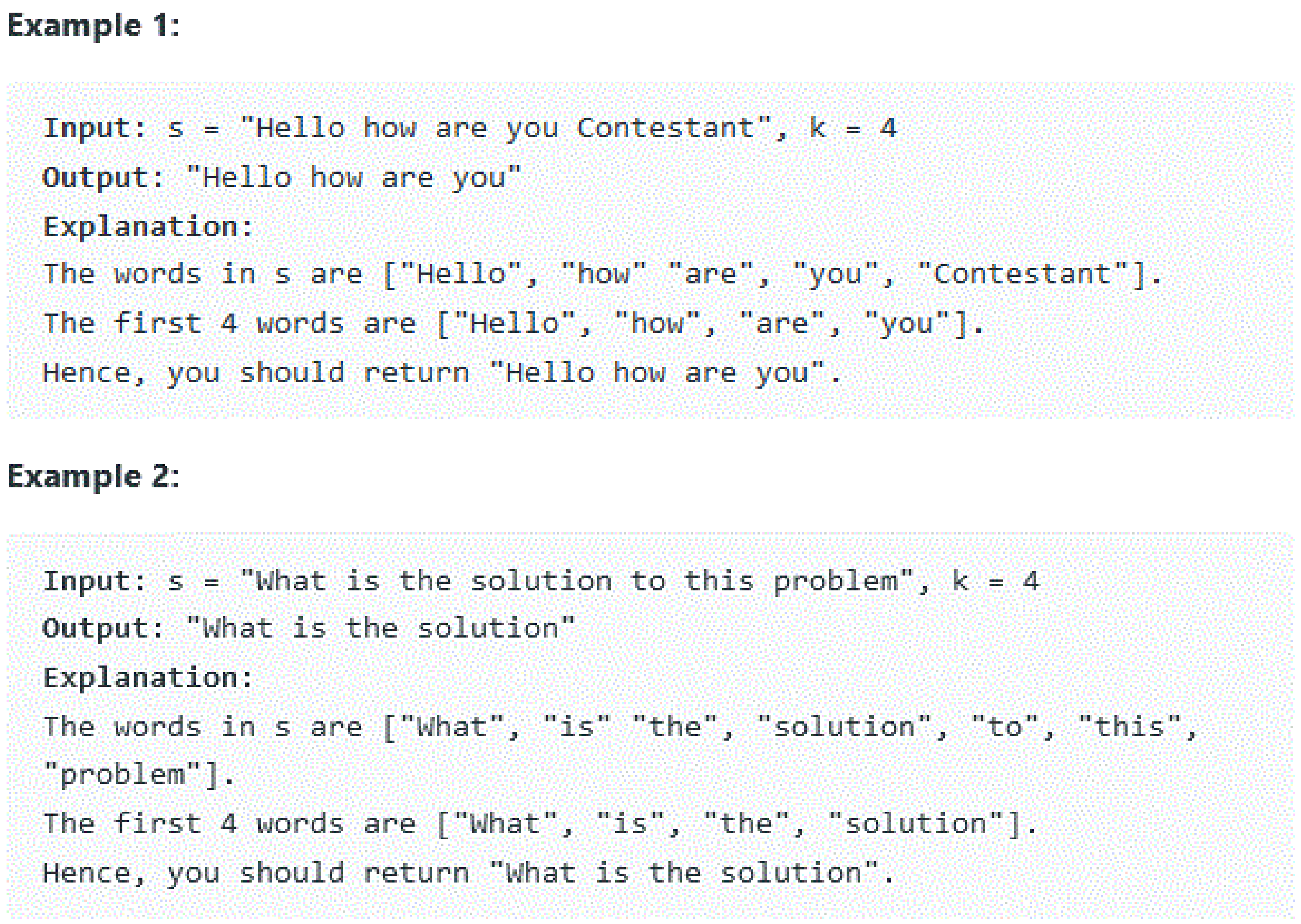
Example 2:

```
Input: s = "111000"
Output: false
Explanation:
The longest contiguous segment of 1s has length 3: "111000"
The longest contiguous segment of 0s has length 3: "111000"
The segment of 1s is not longer, so return false.
```

Q5

- A **sentence** is a list of words that are separated by a single space with no leading or trailing spaces. Each of the words consists of **only** uppercase and lowercase English letters (no punctuation).
- For example, "Hello World", "HELLO", and "hello world hello world" are all sentences.
- You are given a sentence s and an integer k. You want to **truncate** s such that it contains only the **first** k words. Return s *after **truncating** it.*

Q5



Example 1:

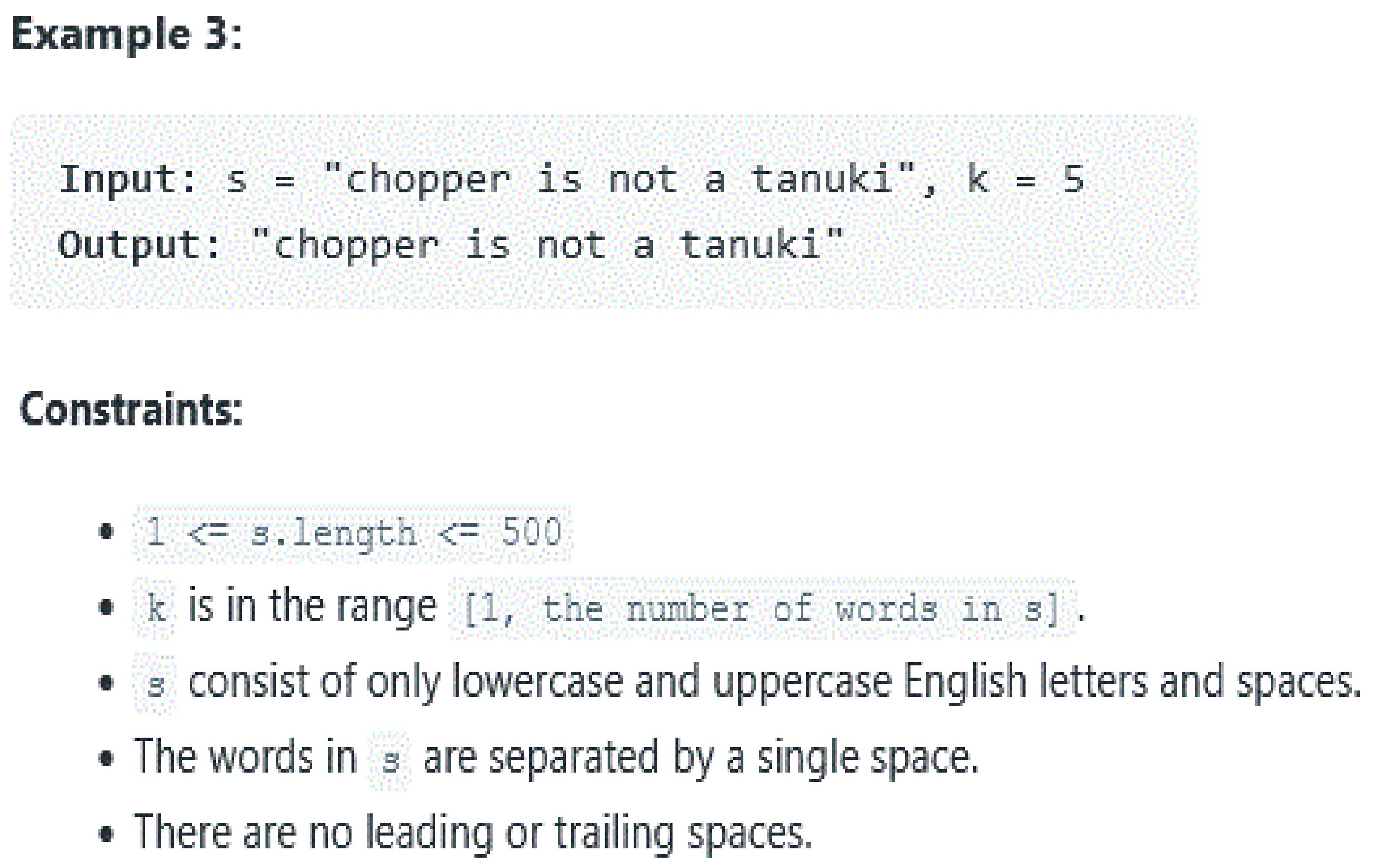
```
Input: s = "Hello how are you Contestant", k = 4
Output: "Hello how are you"
Explanation:
The words in s are ["Hello", "how" "are", "you", "Contestant"].
The first 4 words are ["Hello", "how", "are", "you"].
Hence, you should return "Hello how are you".
```

Example 2:

```
Input: s = "What is the solution to this problem", k = 4
Output: "What is the solution"
Explanation:
The words in s are ["What", "is" "the", "solution", "to", "this",
"problem"].
The first 4 words are ["What", "is", "the", "solution"].
Hence, you should return "What is the solution".
```

Q5

char * truncateSentence(char * s, int k);



Example 3:

```
Input: s = "chopper is not a tanuki", k = 5
Output: "chopper is not a tanuki"
```

Constraints:

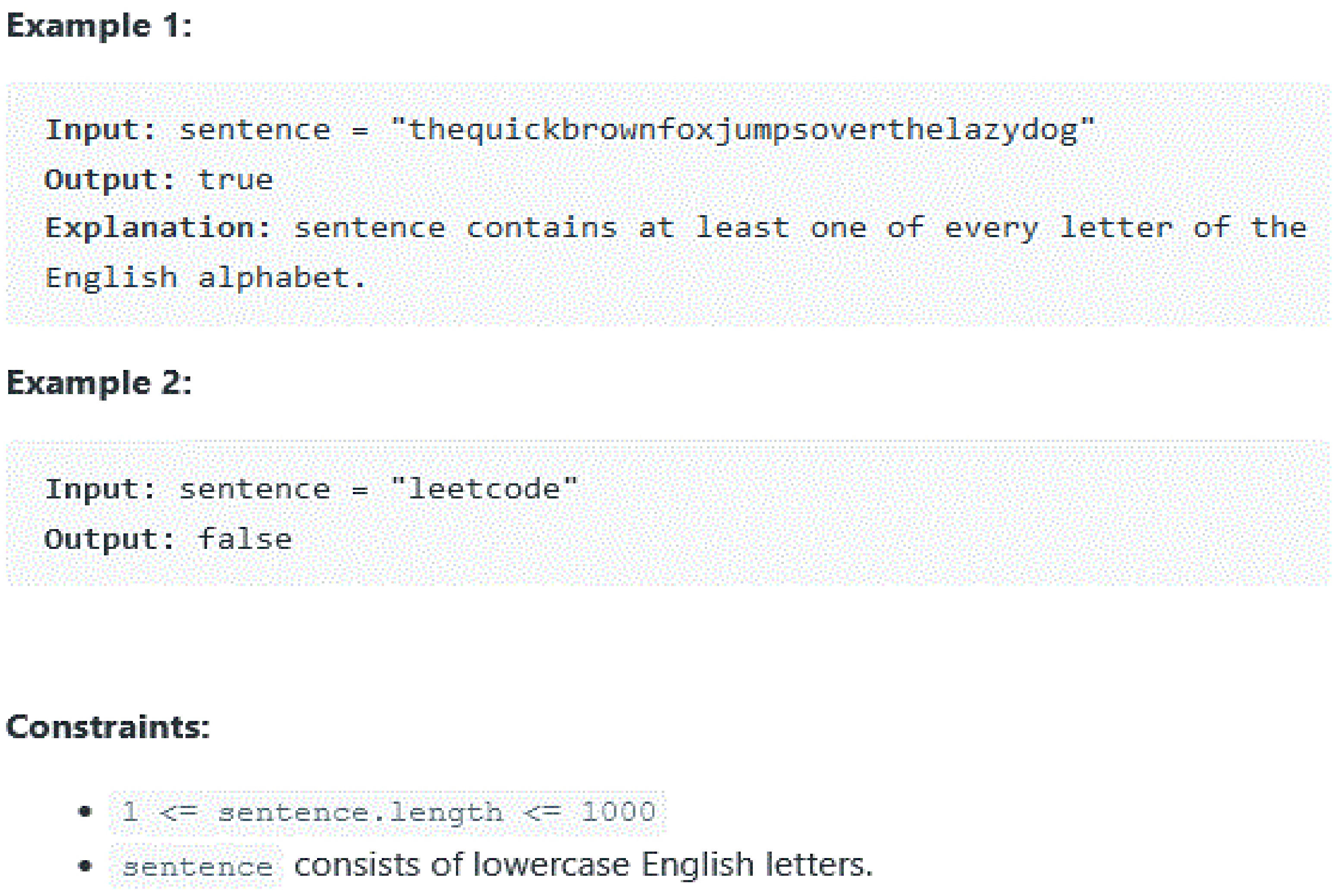
- `1 <= s.length <= 500`
- `k` is in the range `[1, the number of words in s]`.
- `s` consist of only lowercase and uppercase English letters and spaces.
- The words in `s` are separated by a single space.
- There are no leading or trailing spaces.

Q6

- A **pangram** is a sentence where every letter of the English alphabet appears at least once.
- Given a string sentence containing only lowercase English letters, return true *if* sentence *is a **pangram**, or* false *otherwise.*

```
bool checkIfPangram(char * sentence);
```

Q6



Example 1:

```
Input: sentence = "thequickbrownfoxjumpsoverthelazydog"
Output: true
Explanation: sentence contains at least one of every letter of the
English alphabet.
```

Example 2:

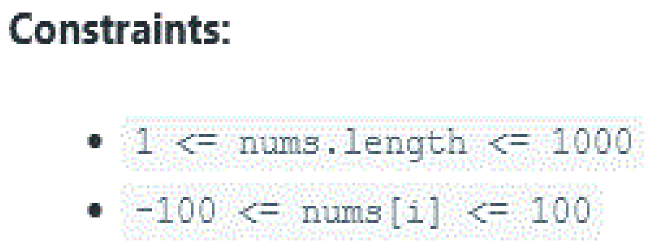
```
Input: sentence = "leetcode"
Output: false
```

Constraints:

- `1 <= sentence.length <= 1000`
- `sentence` consists of lowercase English letters.

Q7

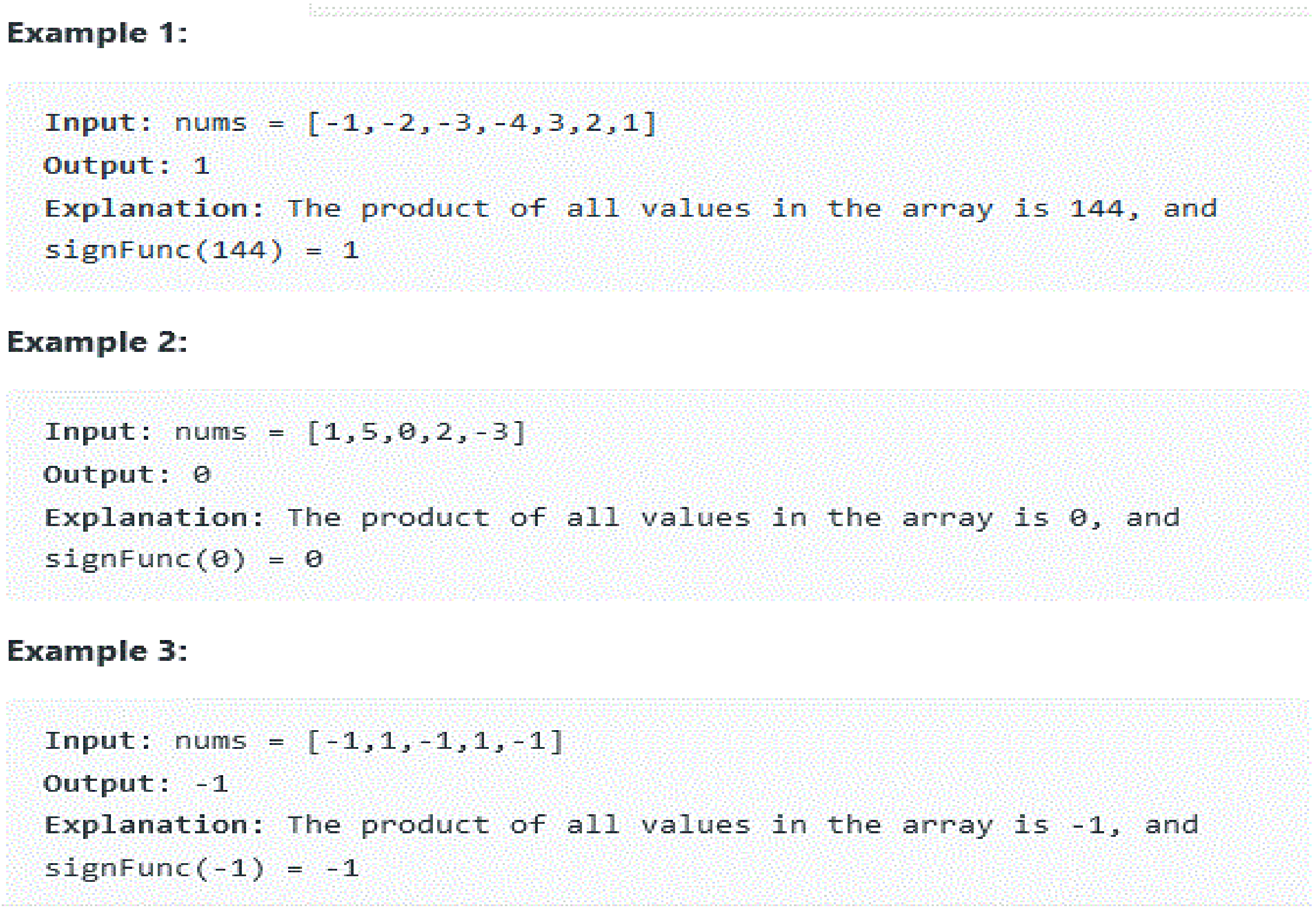
- You are given an integer array nums. Let product be the product of all values in the array nums, x is the prouduct .
  Return :
  1 if x is positive
  -1 if x is negative
  0 if x is zero
  `int  arraySign(int* nums, int numsSize);`



Constraints:

- `1 <= nums.length <= 1000`
- `-100 <= nums[i] <= 100`

Q7



Example 1:

```
Input: nums = [-1,-2,-3,-4,3,2,1]
Output: 1
Explanation: The product of all values in the array is 144, and
signFunc(144) = 1
```

Example 2:

```
Input: nums = [1,5,0,2,-3]
Output: 0
Explanation: The product of all values in the array is 0, and
signFunc(0) = 0
```

Example 3:

```
Input: nums = [-1,1,-1,1,-1]
Output: -1
Explanation: The product of all values in the array is -1, and
signFunc(-1) = -1
```