Poster

Rar the Cat has a sequence of words that he intends to put on a poster. However, the poster has a fixed width that can only fit **W** characters per line (including spaces).

Rar the Cat understands that his sequence of words might not be able to fit on one line of the poster. For example, if **W** = 10, "Rar the Cat jumps over the wall" cannot fit on one line of the poster as it is 31 characters long.

For the poster to look nice, the sequence of words will be arranged on multiple lines in the poster using the following steps:

- Fill in the first line with as many words in the sequence as possible, such that it does not exceed **W** characters.
- Repeat step 1 with the remaining words on the second line, third line, etc…
- Each pair of consecutive words should be separated by a single space. Words should be aligned to the left.
- Words should be arranged in the same order as the provided sequence. No words can be skipped or swapped.

In the above example, when **W** = 10, "Rar the Cat jumps over the wall" will be arranged as such:

```
|Rar the   |
|Cat jumps |
|over the  |
|wall      |
```

In another example, when **W** = 15, "the quick brown fox jumps over the lazy dog" will be arranged as such:

```
|the quick brown|
|fox jumps over |
|the lazy dog   |
```

Rar the Cat feels that arranging the sequence of words manually is too time consuming and hard. Can you code a program to help him do it?

Input
The input will consist of 2 lines.
The first line will be a single integer **W**, the number of characters that can fit into one line of the poster.
The second line will be a sequence of words. The words will be separated by a **single space** character.

Output
The arranged words in the same format as the examples above.
Every line should have exactly **W+2** characters. Print the **W** characters that should be in the poster, with a '|' character immediately before and immediately after the **W** characters.

Limits

- Length of the longest word ≤ **W** ≤ 200
- The second line will be at least 1 and at most 20000 characters long
- The second line will contain between 1 to 2000 words.
- All words will only contain **alphanumeric** characters.

Sample Input (**poster1.in**)

```
10
Rar the Cat jumps over the wall
```

Sample Output (**poster1.out**)

```
|Rar the   |
|Cat jumps |
|over the  |
|wall      |
```

Sample Input (**poster2.in**)

```
15
the quick brown fox jumps over the lazy dog
```

Sample Output (**poster2.out**)

```
|the quick brown|
|fox jumps over |
|the lazy dog   |
```

Notes:

1. You should develop your program in the subdirectory **ex2** and use the skeleton java file provided. You should not create a new file or rename the file provided.
2. You are free to define your own helper methods and classes (or remove existing ones).
3. Please be reminded that the marking scheme is:
   a. Public Test Cases (1%) - 1% for passing **all** test cases, 0% otherwise
   b. Hidden Test Cases (1%) - Partial scoring depending on test cases passed
   c. Manual Grading (1%)
      i. Overall Correctness (correctness of algorithm, severity of bugs)
      ii. Coding Style (meaningful comments, modularity, proper indentation, meaningful method and variable names)
4. Your program will be tested with a time limit of not less than **1 sec** on Codecrunch.

Skeleton File – Poster.java

You are given the below skeleton file `Poster.java`. You should see a non-empty file when you open the skeleton file. Otherwise, you might be in the wrong working directory.

```java
/**
 * Name          :
 * Matric. No    :
 * PLab Acct.    :
 */

import java.util.*;

public class Poster {
    private void run() {
        //implement your "main" method here
    }

    public static void main(String[] args) {
        Poster newPoster = new Poster ();
        newPoster.run();
    }
}
```