Past-paper walk-through

Expressions and Output ▪ 1.3

eXercise

Questions in this set

- 2016 Q4

Reading the mark scheme

- **B1** a mark that stands on its own – the point is either made or not.
- **M1** a *method* mark: the right approach, even if the number is wrong.
- **A1** an *answer* mark, and it usually needs its M mark first.
- **C1** a working mark on the way to the answer.
- // separates answers the examiner accepts equally.
- **ecf** = error carried forward: a later part is marked on your own earlier answer.
- **owtte** = "or words to that effect" – the wording need not match.

Question 4

2016 ▪ Q4

This question involves the process of taking a list of words, called `wordList`, and producing a formatted string of a specified length. The list `wordList` is constructed, spaces are placed in the gaps between words so as many spaces as possible are evenly distributed to each gap, and any leftover spaces are inserted into each gap from left to right until there are no more leftover spaces.

The following three examples illustrate these concepts. In each example, the list of words is to be placed into a formatted string of length 20.

Example 1: wordList: `["AP", "COMP", "SCI", "ROCKS"]`

Question 4

Total number of letters in words: 14 Number of gaps between words: 3 Basic gap width: 2
Leftover spaces: 0
Formatted string:

```
 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
|A |P |  |  |C |O |M |P |  |  |S |C |I |  |  |R |O |C |K |S |
```

Example 2: wordList: `["GREEN", "EGGS", "AND", "HAM"]`

Total number of letters in words: 15 Number of gaps between words: 3 Basic gap width: 1
Leftover spaces: 2

Question 4

The leftover spaces are inserted one at a time between the words from left to right until there are no more extra spaces. In this example, the first two gaps get an extra space.
Formatted string:

```
 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
|G |R |E |E |N |  |  |E |G |G |S |  |  |A |N |D |  |H |A |M |
```

Example 3: wordList: `["BEACH", "BALL"]`

Question 4

Total number of letters in words: 9 Number of gaps between words: 1 Basic gap width: 11 Leftover spaces: 0
Formatted string:

```
 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
|B |E |A |C |H |  |  |  |  |  |  |  |  |  |  |  |B |A |L |L |
```

You will implement three static methods in a class named `StringFormatter` that is not shown.

Question 4

(a) Write the `StringFormatter` method `totalLetters`, which returns the total number of letters in the words in the parameter `wordList`. For example, if the variable `List<String> words` is `["A", "frog", "is"]`, then the call `StringFormatter.totalLetters(words)` returns 7.
Complete method `totalLetters` below.

```
c+cm/** Returns the total number of letters in wordList.
c+cm  *  Precondition: wordList contains at least two words, consisting of letters
c+cm  */
k+kdpublic+w k+kdstatic+w k+ktint+w n+nftotalLettersp(nListonStringo+w nwordListp)
```

Question 4

Class information for this question

```
public class StringFormatter

public static int totalLetters(List<String> wordList)
public static int basicGapWidth(List<String> wordList,
                                int formattedLen)
public static int leftoverSpaces(List<String> wordList,
                                 int formattedLen)
public static String format(List<String> wordList, int formattedLen)
```

Question 4

Class information for this question

```
public class StringFormatter

public static int totalLetters(List<String> wordList)
public static int basicGapWidth(List<String> wordList,
                                int formattedLen)
public static int leftoverSpaces(List<String> wordList,
                                 int formattedLen)
public static String format(List<String> wordList, int formattedLen)
```

Question 4

2016 ▪ Q4

This question involves the process of taking a list of words, called `wordList`, and producing a formatted string of a specified length. The list `wordList` is constructed, spaces are placed in the gaps between words so as many spaces as possible are evenly distributed to each gap, and any leftover spaces are inserted into each gap from left to right until there are no more leftover spaces.

The following three examples illustrate these concepts. In each example, the list of words is to be placed into a formatted string of length 20.

Example 1: wordList: `["AP", "COMP", "SCI", "ROCKS"]`

Question 4

Total number of letters in words: 14 Number of gaps between words: 3 Basic gap width: 2
Leftover spaces: 0
Formatted string:

```
 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
|A |P |  |  |C |O |M |P |  |  |S |C |I |  |  |R |O |C |K |S |
```

Example 2: wordList: `["GREEN", "EGGS", "AND", "HAM"]`

Total number of letters in words: 15 Number of gaps between words: 3 Basic gap width: 1
Leftover spaces: 2

Question 4

The leftover spaces are inserted one at a time between the words from left to right until there are no more extra spaces. In this example, the first two gaps get an extra space.
Formatted string:

```
 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
|G |R |E |E |N |  |  |E |G |G |S |  |  |A |N |D |  |H |A |M |
```

Example 3: wordList: `["BEACH", "BALL"]`

Question 4

Total number of letters in words: 9 Number of gaps between words: 1 Basic gap width: 11
Leftover spaces: 0
Formatted string:

```
 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
|B |E |A |C |H |  |  |  |  |  |  |  |  |  |  |B |A |L |L |
```

You will implement three static methods in a class named `StringFormatter` that is not shown.

(b) Write the `StringFormatter` method `basicGapWidth`, which returns the basic gap width as defined earlier.

Question 4

Class information for this question:

```
public class StringFormatter

public static int totalLetters(List<String> wordList)
public static int basicGapWidth(List<String> wordList,
                                int formattedLen)
public static int leftoverSpaces(List<String> wordList,
                                 int formattedLen)
public static String format(List<String> wordList, int formattedLen)
```

WRITE YOUR SOLUTION ON THE NEXT PAGE.

Assume that `totalLetters` works as specified regardless of what you wrote in part (a). You must use `totalLetters` below.

Question 4

Complete method `basicGapWidth` below.

```
c+cm/** Returns the basic gap width when wordList is used to produce
c+cm  *  a formatted string of formattedLen characters.
c+cm  *  Precondition: wordList contains at least two words, consisting of letters
c+cm  *                formattedLen is large enough for all the words and gaps.
c+cm  */
k+kdpublic+w k+kdstatic+w k+ktint+w n+nfbasicGapWidthp(nListonStringo+w nwordListp
+w                                k+ktint+w nformattedLenp)
```

Question 4

2016 ▪ Q4

This question involves the process of taking a list of words, called `wordList`, and producing a formatted string of a specified length. The list `wordList` is constructed, spaces are placed in the gaps between words so as many spaces as possible are evenly distributed to each gap, and any leftover spaces are inserted into each gap from left to right until there are no more leftover spaces.

The following three examples illustrate these concepts. In each example, the list of words is to be placed into a formatted string of length 20.

Example 1: wordList: `["AP", "COMP", "SCI", "ROCKS"]`

Question 4

Total number of letters in words: 14 Number of gaps between words: 3 Basic gap width: 2
Leftover spaces: 0
Formatted string:

```
 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
|A |P |  |  |C |O |M |P |  |  |S |C |I |  |  |R |O |C |K |S |
```

Example 2: wordList: `["GREEN", "EGGS", "AND", "HAM"]`

Total number of letters in words: 15 Number of gaps between words: 3 Basic gap width: 1
Leftover spaces: 2

Question 4

The leftover spaces are inserted one at a time between the words from left to right until there are no more extra spaces. In this example, the first two gaps get an extra space.
Formatted string:

```
 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
|G |R |E |E |N |  |  |E |G |G |S |  |  |A |N |D |  |H |A |M |
```

Example 3: wordList: `["BEACH", "BALL"]`

Question 4

Total number of letters in words: 9 Number of gaps between words: 1 Basic gap width: 11 Leftover spaces: 0
Formatted string:

```
 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
|B |E |A |C |H |  |  |  |  |  |  |  |  |  |  |  |B |A |L |L |
```

You will implement three static methods in a class named `StringFormatter` that is not shown.

Question 4

(c) Write the `StringFormatter` method `format`, which returns the formatted string as defined earlier. The `StringFormatter` class also contains a method called `leftoverSpaces`, which has already been implemented. This method returns the number of leftover spaces when `wordList` is used to produce a formatted string of `formattedLen` characters.

```
c+cm/** Returns the number of leftover spaces when wordList is used to produce
c+cm  *  a formatted string of formattedLen characters.
c+cm  *  Precondition: wordList contains at least two words, consisting of letters only.
c+cm  *                formattedLen is large enough for all the words and gaps.
c+cm  */
k+kdpublic+w k+kdstatic+w k+ktint+w n+nfleftoverSpacesp(nListonStringo+w nwordListp,
+w                                  k+ktint+w nformattedLenp)
p+w  c+cm/* implementation not shown */+w  p
```

Question 4

Class information for this question:

```
public class StringFormatter

public static int totalLetters(List<String> wordList)
public static int basicGapWidth(List<String> wordList,
                                int formattedLen)
public static int leftoverSpaces(List<String> wordList,
                                 int formattedLen)
public static String format(List<String> wordList, int formattedLen)
```

WRITE YOUR SOLUTION ON THE NEXT PAGE.

Question 4

Assume that `basicGapWidth` works as specified, regardless of what you wrote in part (b). You must use `basicGapWidth` and `leftoverSpaces` appropriately to receive full credit.
Complete method `format` below.

```
c+cm/** Returns a formatted string consisting of the words in wordList separated by spaces
c+cm  *  Precondition: wordList contains at least two words, consisting of letters only.
c+cm  *                formattedLen is large enough for all the words and gaps.
c+cm  *  Postcondition: All words in wordList appear in the formatted string.
c+cm  *                 The words appear in the same order as in wordList.
c+cm  *                 The number of spaces between words is determined by basicGapWidth
c+cm  *                  distribution of leftoverSpaces from left to right, as described i
c+cm  */
k+kdpublic+w k+kdstatic+w nString+w n+nfformatp(nListonStringo+w nwordListp,+w k+ktint+w n
```

Solution 4

(a) Mark scheme

- Model solution (totalLetters — total number of letters across all words in wordList, e.g. ["A","frog","is"] returns 7):
- ```java
- public static int totalLetters(List<String> wordList)
- {
- int total = 0;
- for (String word : wordList)
- {

Solution 4

- total += word.length();
- }
- return total;
- }
- ```
- Scoring (2 pts total): +1 accesses all strings in wordList and adds the length of each to an accumulated count, with no bounds errors; +1 initializes the accumulator correctly (e.g. to 0) and returns the correct final accumulated count.

Solution 4

(b) Mark scheme

- Model solution (basicGapWidth — the minimum number of spaces to place between each pair of words so the formatted string reaches formattedLen characters, computed as leftover space divided evenly among the gaps):
- ‘‘‘java
- public static int basicGapWidth(List<String> wordList, int formattedLen)
- {
- return (formattedLen - totalLetters(wordList)) / (wordList.size() - 1);
- }

Solution 4

- “‘
- Scoring (2 pts total): +1 calls totalLetters (from part a) correctly and uses its result to determine how many total spaces need to be distributed (formattedLen minus total letters); +1 returns the correct calculated value, dividing the leftover space by the number of gaps (wordList.size() - 1).

Solution 4

(c) Mark scheme

- Model solution (format — builds the formatted string of length formattedLen from wordList, distributing basicGapWidth spaces in every gap plus one extra space in the first leftoverSpaces gaps; leftoverSpaces is a given, already-implemented helper):
- “‘java
- public static String format(List<String> wordList, int formattedLen)
- {
- String formatted = ””;
- int gapWidth = basicGapWidth(wordList, formattedLen);

Solution 4

- int leftovers = leftoverSpaces(wordList, formattedLen);
- for (int w = 0; w < wordList.size() - 1; w++)
- {
- formatted = formatted + wordList.get(w);
- for (int i = 0; i < gapWidth; i++)
- {
- formatted = formatted + " ";
- }
- if (leftovers > 0)
- {

Solution 4

- formatted = formatted + " ";
- leftovers–;
- }
- }
- formatted = formatted + wordList.get(wordList.size() - 1);
- return formatted;
- }
- ‘‘‘

Solution 4

- Scoring (5 pts total): $+1$ calls basicGapWidth and leftoverSpaces correctly and uses their results; $+1$ adds all strings in wordList to the formatted string in their original order, with no bounds errors; $+1$ inserts basicGapWidth spaces between each pair of words in the formatted string; $+1$ inserts one additional space between the first leftoverSpaces pairs of words (distributing the leftover spaces across the earliest gaps); $+1$ initializes and returns the formatted string correctly, with no extra or deleted characters (i.e. the final length matches formattedLen exactly).