[PT] modularização

J - Soma tripla

Faça um programa que, leia um número inteiro positivo N (N<=100) e visualize todas as suas decomposições distintas como soma de três inteiros positivos.
As triplas que possuam os mesmos valores, mas por ordem diferente, devem ser ignoradas. No final deve ainda visualizar a quantidade de triplas distintas calculadas.
Os valores maiores devem aparecer à esquerda.
Apresente a informação no seguinte formato:

<n1> + <n2> + <n3>
<n4> + <n5> + <n6>
...
triples=<triplas distintas>

Implemente um método que calcule, visualize as triplas e retorne a quantidade de triplas distintas.

[EN] modularization

J - Triple Sum

Write a program that reads a positive integer N (N<=100) and displays all of its distinct decompositions as the sum of three positive integers.
Triples that have the same values, but in different order, should be ignored. At the end, you should also display the number of distinct triples.
Larger values should appear on the left.
Display the information in the following format:

<n1> + <n2> + <n3>
<n4> + <n5> + <n6>
...
triples=<distinct triples>

Implement a method that calculates, displays the triples and returns the number of distinct triples.

Example1:

Input	Output
5	3 + 1 + 1 2 + 2 + 1 triples=2

Example2:

Input	Output
9	7 + 1 + 1 6 + 2 + 1 5 + 3 + 1 5 + 2 + 2 4 + 4 + 1 4 + 3 + 2 3 + 3 + 3 triples=7

Example3:

Input	Output
23	21 + 1 + 1 20 + 2 + 1 19 + 3 + 1 19 + 2 + 2 18 + 4 + 1 18 + 3 + 2 17 + 5 + 1 17 + 4 + 2 17 + 3 + 3 16 + 6 + 1 16 + 5 + 2 16 + 4 + 3 15 + 7 + 1 15 + 6 + 2 15 + 5 + 3 15 + 4 + 4 14 + 8 + 1 14 + 7 + 2 14 + 6 + 3 14 + 5 + 4

	13 + 9 + 1 13 + 8 + 2 13 + 7 + 3 13 + 6 + 4 13 + 5 + 5 12 + 10 + 1 12 + 9 + 2 12 + 8 + 3 12 + 7 + 4 12 + 6 + 5 11 + 11 + 1 11 + 10 + 2 11 + 9 + 3 11 + 8 + 4 11 + 7 + 5 11 + 6 + 6 10 + 10 + 3 10 + 9 + 4 10 + 8 + 5 10 + 7 + 6 9 + 9 + 5 9 + 8 + 6 9 + 7 + 7 8 + 8 + 7 triples=44

Example4:

Input	Output
100	98 + 1 + 1 97 + 2 + 1 96 + 3 + 1 96 + 2 + 2 95 + 4 + 1 95 + 3 + 2 94 + 5 + 1 94 + 4 + 2 94 + 3 + 3 93 + 6 + 1 93 + 5 + 2 93 + 4 + 3 92 + 7 + 1 92 + 6 + 2 92 + 5 + 3 92 + 4 + 4 91 + 8 + 1 91 + 7 + 2 91 + 6 + 3 91 + 5 + 4 90 + 9 + 1 90 + 8 + 2 90 + 7 + 3 90 + 6 + 4 90 + 5 + 5 89 + 10 + 1 89 + 9 + 2 89 + 8 + 3 89 + 7 + 4 89 + 6 + 5 88 + 11 + 1 88 + 10 + 2 88 + 9 + 3 88 + 8 + 4 88 + 7 + 5 88 + 6 + 6 87 + 12 + 1 87 + 11 + 2 87 + 10 + 3 87 + 9 + 4 87 + 8 + 5 87 + 7 + 6 86 + 13 + 1 86 + 12 + 2 86 + 11 + 3 86 + 10 + 4 86 + 9 + 5 86 + 8 + 6 86 + 7 + 7 85 + 14 + 1 85 + 13 + 2 85 + 12 + 3 85 + 11 + 4 85 + 10 + 5 85 + 9 + 6 85 + 8 + 7 84 + 15 + 1

```
84 + 14 + 2
84 + 13 + 3
84 + 12 + 4
84 + 11 + 5
84 + 10 + 6
84 + 9 + 7
84 + 8 + 8
83 + 16 + 1
83 + 15 + 2
83 + 14 + 3
83 + 13 + 4
83 + 12 + 5
83 + 11 + 6
83 + 10 + 7
83 + 9 + 8
82 + 17 + 1
82 + 16 + 2
82 + 15 + 3
82 + 14 + 4
82 + 13 + 5
82 + 12 + 6
82 + 11 + 7
82 + 10 + 8
82 + 9 + 9
81 + 18 + 1
81 + 17 + 2
81 + 16 + 3
81 + 15 + 4
81 + 14 + 5
81 + 13 + 6
81 + 12 + 7
81 + 11 + 8
81 + 10 + 9
80 + 19 + 1
80 + 18 + 2
80 + 17 + 3
80 + 16 + 4
80 + 15 + 5
80 + 14 + 6
80 + 13 + 7
80 + 12 + 8
80 + 11 + 9
80 + 10 + 10
79 + 20 + 1
79 + 19 + 2
79 + 18 + 3
79 + 17 + 4
79 + 16 + 5
79 + 15 + 6
79 + 14 + 7
79 + 13 + 8
79 + 12 + 9
79 + 11 + 10
78 + 21 + 1
78 + 20 + 2
78 + 19 + 3
78 + 18 + 4
78 + 17 + 5
78 + 16 + 6
78 + 15 + 7
78 + 14 + 8
78 + 13 + 9
78 + 12 + 10
78 + 11 + 11
77 + 22 + 1
77 + 21 + 2
77 + 20 + 3
77 + 19 + 4
77 + 18 + 5
77 + 17 + 6
77 + 16 + 7
77 + 15 + 8
77 + 14 + 9
77 + 13 + 10
77 + 12 + 11
76 + 23 + 1
76 + 22 + 2
76 + 21 + 3
76 + 20 + 4
76 + 19 + 5
76 + 18 + 6
76 + 17 + 7
76 + 16 + 8
76 + 15 + 9
76 + 14 + 10
76 + 13 + 11
76 + 12 + 12
```

```
75 + 24 + 1
75 + 23 + 2
75 + 22 + 3
75 + 21 + 4
75 + 20 + 5
75 + 19 + 6
75 + 18 + 7
75 + 17 + 8
75 + 16 + 9
75 + 15 + 10
75 + 14 + 11
75 + 13 + 12
74 + 25 + 1
74 + 24 + 2
74 + 23 + 3
74 + 22 + 4
74 + 21 + 5
74 + 20 + 6
74 + 19 + 7
74 + 18 + 8
74 + 17 + 9
74 + 16 + 10
74 + 15 + 11
74 + 14 + 12
74 + 13 + 13
73 + 26 + 1
73 + 25 + 2
73 + 24 + 3
73 + 23 + 4
73 + 22 + 5
73 + 21 + 6
73 + 20 + 7
73 + 19 + 8
73 + 18 + 9
73 + 17 + 10
73 + 16 + 11
73 + 15 + 12
73 + 14 + 13
72 + 27 + 1
72 + 26 + 2
72 + 25 + 3
72 + 24 + 4
72 + 23 + 5
72 + 22 + 6
72 + 21 + 7
72 + 20 + 8
72 + 19 + 9
72 + 18 + 10
72 + 17 + 11
72 + 16 + 12
72 + 15 + 13
72 + 14 + 14
71 + 28 + 1
71 + 27 + 2
71 + 26 + 3
71 + 25 + 4
71 + 24 + 5
71 + 23 + 6
71 + 22 + 7
71 + 21 + 8
71 + 20 + 9
71 + 19 + 10
71 + 18 + 11
71 + 17 + 12
71 + 16 + 13
71 + 15 + 14
70 + 29 + 1
70 + 28 + 2
70 + 27 + 3
70 + 26 + 4
70 + 25 + 5
70 + 24 + 6
70 + 23 + 7
70 + 22 + 8
70 + 21 + 9
70 + 20 + 10
70 + 19 + 11
70 + 18 + 12
70 + 17 + 13
70 + 16 + 14
70 + 15 + 15
69 + 30 + 1
69 + 29 + 2
69 + 28 + 3
69 + 27 + 4
69 + 26 + 5
69 + 25 + 6
```

```
69 + 24 + 7
69 + 23 + 8
69 + 22 + 9
69 + 21 + 10
69 + 20 + 11
69 + 19 + 12
69 + 18 + 13
69 + 17 + 14
69 + 16 + 15
68 + 31 + 1
68 + 30 + 2
68 + 29 + 3
68 + 28 + 4
68 + 27 + 5
68 + 26 + 6
68 + 25 + 7
68 + 24 + 8
68 + 23 + 9
68 + 22 + 10
68 + 21 + 11
68 + 20 + 12
68 + 19 + 13
68 + 18 + 14
68 + 17 + 15
68 + 16 + 16
67 + 32 + 1
67 + 31 + 2
67 + 30 + 3
67 + 29 + 4
67 + 28 + 5
67 + 27 + 6
67 + 26 + 7
67 + 25 + 8
67 + 24 + 9
67 + 23 + 10
67 + 22 + 11
67 + 21 + 12
67 + 20 + 13
67 + 19 + 14
67 + 18 + 15
67 + 17 + 16
66 + 33 + 1
66 + 32 + 2
66 + 31 + 3
66 + 30 + 4
66 + 29 + 5
66 + 28 + 6
66 + 27 + 7
66 + 26 + 8
66 + 25 + 9
66 + 24 + 10
66 + 23 + 11
66 + 22 + 12
66 + 21 + 13
66 + 20 + 14
66 + 19 + 15
66 + 18 + 16
66 + 17 + 17
65 + 34 + 1
65 + 33 + 2
65 + 32 + 3
65 + 31 + 4
65 + 30 + 5
65 + 29 + 6
65 + 28 + 7
65 + 27 + 8
65 + 26 + 9
65 + 25 + 10
65 + 24 + 11
65 + 23 + 12
65 + 22 + 13
65 + 21 + 14
65 + 20 + 15
65 + 19 + 16
65 + 18 + 17
64 + 35 + 1
64 + 34 + 2
64 + 33 + 3
64 + 32 + 4
64 + 31 + 5
64 + 30 + 6
64 + 29 + 7
64 + 28 + 8
64 + 27 + 9
64 + 26 + 10
64 + 25 + 11
64 + 24 + 12
```

```
64 + 23 + 13
64 + 22 + 14
64 + 21 + 15
64 + 20 + 16
64 + 19 + 17
64 + 18 + 18
63 + 36 + 1
63 + 35 + 2
63 + 34 + 3
63 + 33 + 4
63 + 32 + 5
63 + 31 + 6
63 + 30 + 7
63 + 29 + 8
63 + 28 + 9
63 + 27 + 10
63 + 26 + 11
63 + 25 + 12
63 + 24 + 13
63 + 23 + 14
63 + 22 + 15
63 + 21 + 16
63 + 20 + 17
63 + 19 + 18
62 + 37 + 1
62 + 36 + 2
62 + 35 + 3
62 + 34 + 4
62 + 33 + 5
62 + 32 + 6
62 + 31 + 7
62 + 30 + 8
62 + 29 + 9
62 + 28 + 10
62 + 27 + 11
62 + 26 + 12
62 + 25 + 13
62 + 24 + 14
62 + 23 + 15
62 + 22 + 16
62 + 21 + 17
62 + 20 + 18
62 + 19 + 19
61 + 38 + 1
61 + 37 + 2
61 + 36 + 3
61 + 35 + 4
61 + 34 + 5
61 + 33 + 6
61 + 32 + 7
61 + 31 + 8
61 + 30 + 9
61 + 29 + 10
61 + 28 + 11
61 + 27 + 12
61 + 26 + 13
61 + 25 + 14
61 + 24 + 15
61 + 23 + 16
61 + 22 + 17
61 + 21 + 18
61 + 20 + 19
60 + 39 + 1
60 + 38 + 2
60 + 37 + 3
60 + 36 + 4
60 + 35 + 5
60 + 34 + 6
60 + 33 + 7
60 + 32 + 8
60 + 31 + 9
60 + 30 + 10
60 + 29 + 11
60 + 28 + 12
60 + 27 + 13
60 + 26 + 14
60 + 25 + 15
60 + 24 + 16
60 + 23 + 17
60 + 22 + 18
60 + 21 + 19
60 + 20 + 20
59 + 40 + 1
59 + 39 + 2
59 + 38 + 3
59 + 37 + 4
59 + 36 + 5
```

```
59 + 35 + 6
59 + 34 + 7
59 + 33 + 8
59 + 32 + 9
59 + 31 + 10
59 + 30 + 11
59 + 29 + 12
59 + 28 + 13
59 + 27 + 14
59 + 26 + 15
59 + 25 + 16
59 + 24 + 17
59 + 23 + 18
59 + 22 + 19
59 + 21 + 20
58 + 41 + 1
58 + 40 + 2
58 + 39 + 3
58 + 38 + 4
58 + 37 + 5
58 + 36 + 6
58 + 35 + 7
58 + 34 + 8
58 + 33 + 9
58 + 32 + 10
58 + 31 + 11
58 + 30 + 12
58 + 29 + 13
58 + 28 + 14
58 + 27 + 15
58 + 26 + 16
58 + 25 + 17
58 + 24 + 18
58 + 23 + 19
58 + 22 + 20
58 + 21 + 21
57 + 42 + 1
57 + 41 + 2
57 + 40 + 3
57 + 39 + 4
57 + 38 + 5
57 + 37 + 6
57 + 36 + 7
57 + 35 + 8
57 + 34 + 9
57 + 33 + 10
57 + 32 + 11
57 + 31 + 12
57 + 30 + 13
57 + 29 + 14
57 + 28 + 15
57 + 27 + 16
57 + 26 + 17
57 + 25 + 18
57 + 24 + 19
57 + 23 + 20
57 + 22 + 21
56 + 43 + 1
56 + 42 + 2
56 + 41 + 3
56 + 40 + 4
56 + 39 + 5
56 + 38 + 6
56 + 37 + 7
56 + 36 + 8
56 + 35 + 9
56 + 34 + 10
56 + 33 + 11
56 + 32 + 12
56 + 31 + 13
56 + 30 + 14
56 + 29 + 15
56 + 28 + 16
56 + 27 + 17
56 + 26 + 18
56 + 25 + 19
56 + 24 + 20
56 + 23 + 21
56 + 22 + 22
55 + 44 + 1
55 + 43 + 2
55 + 42 + 3
55 + 41 + 4
55 + 40 + 5
55 + 39 + 6
55 + 38 + 7
55 + 37 + 8
```

```
55 + 36 + 9
55 + 35 + 10
55 + 34 + 11
55 + 33 + 12
55 + 32 + 13
55 + 31 + 14
55 + 30 + 15
55 + 29 + 16
55 + 28 + 17
55 + 27 + 18
55 + 26 + 19
55 + 25 + 20
55 + 24 + 21
55 + 23 + 22
54 + 45 + 1
54 + 44 + 2
54 + 43 + 3
54 + 42 + 4
54 + 41 + 5
54 + 40 + 6
54 + 39 + 7
54 + 38 + 8
54 + 37 + 9
54 + 36 + 10
54 + 35 + 11
54 + 34 + 12
54 + 33 + 13
54 + 32 + 14
54 + 31 + 15
54 + 30 + 16
54 + 29 + 17
54 + 28 + 18
54 + 27 + 19
54 + 26 + 20
54 + 25 + 21
54 + 24 + 22
54 + 23 + 23
53 + 46 + 1
53 + 45 + 2
53 + 44 + 3
53 + 43 + 4
53 + 42 + 5
53 + 41 + 6
53 + 40 + 7
53 + 39 + 8
53 + 38 + 9
53 + 37 + 10
53 + 36 + 11
53 + 35 + 12
53 + 34 + 13
53 + 33 + 14
53 + 32 + 15
53 + 31 + 16
53 + 30 + 17
53 + 29 + 18
53 + 28 + 19
53 + 27 + 20
53 + 26 + 21
53 + 25 + 22
53 + 24 + 23
52 + 47 + 1
52 + 46 + 2
52 + 45 + 3
52 + 44 + 4
52 + 43 + 5
52 + 42 + 6
52 + 41 + 7
52 + 40 + 8
52 + 39 + 9
52 + 38 + 10
52 + 37 + 11
52 + 36 + 12
52 + 35 + 13
52 + 34 + 14
52 + 33 + 15
52 + 32 + 16
52 + 31 + 17
52 + 30 + 18
52 + 29 + 19
52 + 28 + 20
52 + 27 + 21
52 + 26 + 22
52 + 25 + 23
52 + 24 + 24
51 + 48 + 1
51 + 47 + 2
51 + 46 + 3
```

```
51 + 45 + 4
51 + 44 + 5
51 + 43 + 6
51 + 42 + 7
51 + 41 + 8
51 + 40 + 9
51 + 39 + 10
51 + 38 + 11
51 + 37 + 12
51 + 36 + 13
51 + 35 + 14
51 + 34 + 15
51 + 33 + 16
51 + 32 + 17
51 + 31 + 18
51 + 30 + 19
51 + 29 + 20
51 + 28 + 21
51 + 27 + 22
51 + 26 + 23
51 + 25 + 24
50 + 49 + 1
50 + 48 + 2
50 + 47 + 3
50 + 46 + 4
50 + 45 + 5
50 + 44 + 6
50 + 43 + 7
50 + 42 + 8
50 + 41 + 9
50 + 40 + 10
50 + 39 + 11
50 + 38 + 12
50 + 37 + 13
50 + 36 + 14
50 + 35 + 15
50 + 34 + 16
50 + 33 + 17
50 + 32 + 18
50 + 31 + 19
50 + 30 + 20
50 + 29 + 21
50 + 28 + 22
50 + 27 + 23
50 + 26 + 24
50 + 25 + 25
49 + 49 + 2
49 + 48 + 3
49 + 47 + 4
49 + 46 + 5
49 + 45 + 6
49 + 44 + 7
49 + 43 + 8
49 + 42 + 9
49 + 41 + 10
49 + 40 + 11
49 + 39 + 12
49 + 38 + 13
49 + 37 + 14
49 + 36 + 15
49 + 35 + 16
49 + 34 + 17
49 + 33 + 18
49 + 32 + 19
49 + 31 + 20
49 + 30 + 21
49 + 29 + 22
49 + 28 + 23
49 + 27 + 24
49 + 26 + 25
48 + 48 + 4
48 + 47 + 5
48 + 46 + 6
48 + 45 + 7
48 + 44 + 8
48 + 43 + 9
48 + 42 + 10
48 + 41 + 11
48 + 40 + 12
48 + 39 + 13
48 + 38 + 14
48 + 37 + 15
48 + 36 + 16
48 + 35 + 17
48 + 34 + 18
48 + 33 + 19
48 + 32 + 20
```

```
48 + 31 + 21
48 + 30 + 22
48 + 29 + 23
48 + 28 + 24
48 + 27 + 25
48 + 26 + 26
47 + 47 + 6
47 + 46 + 7
47 + 45 + 8
47 + 44 + 9
47 + 43 + 10
47 + 42 + 11
47 + 41 + 12
47 + 40 + 13
47 + 39 + 14
47 + 38 + 15
47 + 37 + 16
47 + 36 + 17
47 + 35 + 18
47 + 34 + 19
47 + 33 + 20
47 + 32 + 21
47 + 31 + 22
47 + 30 + 23
47 + 29 + 24
47 + 28 + 25
47 + 27 + 26
46 + 46 + 8
46 + 45 + 9
46 + 44 + 10
46 + 43 + 11
46 + 42 + 12
46 + 41 + 13
46 + 40 + 14
46 + 39 + 15
46 + 38 + 16
46 + 37 + 17
46 + 36 + 18
46 + 35 + 19
46 + 34 + 20
46 + 33 + 21
46 + 32 + 22
46 + 31 + 23
46 + 30 + 24
46 + 29 + 25
46 + 28 + 26
46 + 27 + 27
45 + 45 + 10
45 + 44 + 11
45 + 43 + 12
45 + 42 + 13
45 + 41 + 14
45 + 40 + 15
45 + 39 + 16
45 + 38 + 17
45 + 37 + 18
45 + 36 + 19
45 + 35 + 20
45 + 34 + 21
45 + 33 + 22
45 + 32 + 23
45 + 31 + 24
45 + 30 + 25
45 + 29 + 26
45 + 28 + 27
44 + 44 + 12
44 + 43 + 13
44 + 42 + 14
44 + 41 + 15
44 + 40 + 16
44 + 39 + 17
44 + 38 + 18
44 + 37 + 19
44 + 36 + 20
44 + 35 + 21
44 + 34 + 22
44 + 33 + 23
44 + 32 + 24
44 + 31 + 25
44 + 30 + 26
44 + 29 + 27
44 + 28 + 28
43 + 43 + 14
43 + 42 + 15
43 + 41 + 16
43 + 40 + 17
43 + 39 + 18
```

```
43 + 38 + 19
43 + 37 + 20
43 + 36 + 21
43 + 35 + 22
43 + 34 + 23
43 + 33 + 24
43 + 32 + 25
43 + 31 + 26
43 + 30 + 27
43 + 29 + 28
42 + 42 + 16
42 + 41 + 17
42 + 40 + 18
42 + 39 + 19
42 + 38 + 20
42 + 37 + 21
42 + 36 + 22
42 + 35 + 23
42 + 34 + 24
42 + 33 + 25
42 + 32 + 26
42 + 31 + 27
42 + 30 + 28
42 + 29 + 29
41 + 41 + 18
41 + 40 + 19
41 + 39 + 20
41 + 38 + 21
41 + 37 + 22
41 + 36 + 23
41 + 35 + 24
41 + 34 + 25
41 + 33 + 26
41 + 32 + 27
41 + 31 + 28
41 + 30 + 29
40 + 40 + 20
40 + 39 + 21
40 + 38 + 22
40 + 37 + 23
40 + 36 + 24
40 + 35 + 25
40 + 34 + 26
40 + 33 + 27
40 + 32 + 28
40 + 31 + 29
40 + 30 + 30
39 + 39 + 22
39 + 38 + 23
39 + 37 + 24
39 + 36 + 25
39 + 35 + 26
39 + 34 + 27
39 + 33 + 28
39 + 32 + 29
39 + 31 + 30
38 + 38 + 24
38 + 37 + 25
38 + 36 + 26
38 + 35 + 27
38 + 34 + 28
38 + 33 + 29
38 + 32 + 30
38 + 31 + 31
37 + 37 + 26
37 + 36 + 27
37 + 35 + 28
37 + 34 + 29
37 + 33 + 30
37 + 32 + 31
36 + 36 + 28
36 + 35 + 29
36 + 34 + 30
36 + 33 + 31
36 + 32 + 32
35 + 35 + 30
35 + 34 + 31
35 + 33 + 32
34 + 34 + 32
34 + 33 + 33
triples=833
```

[PT] modularização

K - Sorte ou azar aos dados

Faça um programa que simule um jogo de dados em que um jogador lança um dado um conjunto de vezes. A pontuação do jogador é a soma do valor das faces do dado. As faces do dado são numeradas de 1 a 6. Se saírem duas faces iguais consecutivas, a pontuação passa a negativo. Quando uma face é igual à anterior, os pontos da segunda face são negativos, caso contrário são positivos.
O programa deve ler um número inteiro positivo, cujos algarismos representam o valor das faces que saíram, e visualizar a pontuação final.

Implemente um método que receba um número inteiro e retorne a pontuação de acordo com as regras do jogo.

[EN] modularization

K - Lucky or Unlucky to the dice

Write a program that simulates a game of dice in which a player toss a die a set of times. The player's score is the sum of the faces alue of the die. The faces of the die are numbered from 1 to 6. If two consecutive equal faces come out, the score becomes negative. When one face is the same as the previous one, the points of the second face are negative, otherwise they are positive.
The program must read a positive integer, whose digits represent the consecutive value of the faces, and display the final score.

Implement a method that takes an integer as a parameter and returns the corresponding score, according to the rules of the game.

examples:
3165: 3 > (3) + 1 > (4) + 6 > (10) + 5 > (15)
344: 3 > (3) + 4 > (7) + 4 > (-7) - 4 > (-11)
34414: 3 > (3) + 4 > (7) + 4 > (-7) - 4 > (-11) + 1 > (-10) + 4 > (-6)

Example1:

Input	Output
3165	points=15

Example2:

Input	Output
344	points=-11

Example3:

Input	Output
34414	points=-6

Example4:

Input	Output
343313311	points=-12

Example5:

Input	Output
4444	points=-16

Example6:

Input	Output
1234321	points=16

[PT] modularização

L - Quantidade de palavras de uma frase

Elabore um programa que leia uma frase e escreva quantas palavras possui essa frase.
Considera-se uma palavra um qualquer conjunto de símbolos que se encontram entre espaços.

Implemente um método que recebe a frase e retorna a quantidade de palavras existentes nessa frase.

[EN] modularization

L - Amount of words

Write a program to read a sentence and to write how many words the sentence has.
A word is considered to be any set of symbols that are between spaces.

Implement a module that receives the sentence and returns the amount of words in that phrase.

Example1:

Input	Output
Olá bom dia.	3

Example2:

Input	Output
quem sou eu ?	4

Example3:

Input	Output
9 semanas, e 1/2 !	5

Example4:

Input	Output
	0

[PT] modularização

Y - Tabuadas de um intervalo

Faça um programa que permita escrever tabuadas dos números inteiros positivos pertencentes a um intervalo fechado definido pelo utilizador.

Implemente o programa usando:
a) Um método (readPositiveValue) que leia e retorne um número inteiro positivo maior que zero. Devem ser lidos números continuamente até ser introduzido um número válido. O método retorna o número validado.

b) Um método (showTablesInRange) que recebe por parâmetro os limites do intervalo e tem a responsabilidade de processar a tabuada de todos os números desse intervalo. As tabuadas devem ser processadas por ordem crescente.

c) Um método (showTableOfNumber) que escreve a tabuada de um número específico recebido como parâmetro. O resultado deve seguir o seguinte formato (exemplo para a tabuada de 7):

```
Multiplication table of 7
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
7 x 4 = 28
7 x 5 = 35
7 x 6 = 42
7 x 7 = 49
7 x 8 = 56
7 x 9 = 63
7 x 10 = 70
```

[EN] modularization

Y - Multiplication tables

Write a program to display multiplication tables of numbers between a user-defined closed range.

The program should read two positive integers that define the range boundaries. Any negative number should be ignored. Then, the multiplication tables must be displayed in ascending order.

Implement the program using:
a) A method (readPositiveValue) to read and return a positive integer greater than zero. Numbers should be read continuously until a valid number is entered. The method returns the validated number;

b) A method (showTablesInRange) to receive the range limits as a parameter and has the responsibility to process the multiplication table of all numbers in that range;

c) A method (showTableOfNumber) to display the multiplication table of a specific number received as a parameter. The result should follow the following format (example for the multiplication table of 7):

```
Multiplication table of 7
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
7 x 4 = 28
7 x 5 = 35
7 x 6 = 42
7 x 7 = 49
7 x 8 = 56
7 x 9 = 63
7 x 10 = 70
```

Example1:

Input	Output
8 7	Multiplication table of 7 7 x 1 = 7 7 x 2 = 14 7 x 3 = 21 7 x 4 = 28 7 x 5 = 35 7 x 6 = 42 7 x 7 = 49 7 x 8 = 56 7 x 9 = 63 7 x 10 = 70 Multiplication table of 8 8 x 1 = 8 8 x 2 = 16 8 x 3 = 24 8 x 4 = 32 8 x 5 = 40 8 x 6 = 48 8 x 7 = 56 8 x 8 = 64 8 x 9 = 72 8 x 10 = 80

Example2:

Input	Output
-2 3 -5 -6 3	Multiplication table of 3 3 x 1 = 3 3 x 2 = 6 3 x 3 = 9 3 x 4 = 12 3 x 5 = 15 3 x 6 = 18 3 x 7 = 21 3 x 8 = 24 3 x 9 = 27 3 x 10 = 30

Example3:

Input	Output
-1 -2 0 1 0 10	Multiplication table of 1 1 x 1 = 1 1 x 2 = 2 1 x 3 = 3 1 x 4 = 4 1 x 5 = 5 1 x 6 = 6 1 x 7 = 7 1 x 8 = 8 1 x 9 = 9 1 x 10 = 10 Multiplication table of 2 2 x 1 = 2 2 x 2 = 4 2 x 3 = 6 2 x 4 = 8 2 x 5 = 10 2 x 6 = 12 2 x 7 = 14 2 x 8 = 16 2 x 9 = 18 2 x 10 = 20 Multiplication table of 3

```
3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
3 x 10 = 30
Multiplication table of 4
4 x 1 = 4
4 x 2 = 8
4 x 3 = 12
4 x 4 = 16
4 x 5 = 20
4 x 6 = 24
4 x 7 = 28
4 x 8 = 32
4 x 9 = 36
4 x 10 = 40
Multiplication table of 5
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
Multiplication table of 6
6 x 1 = 6
6 x 2 = 12
6 x 3 = 18
6 x 4 = 24
6 x 5 = 30
6 x 6 = 36
6 x 7 = 42
6 x 8 = 48
6 x 9 = 54
6 x 10 = 60
Multiplication table of 7
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
7 x 4 = 28
7 x 5 = 35
7 x 6 = 42
7 x 7 = 49
7 x 8 = 56
7 x 9 = 63
7 x 10 = 70
Multiplication table of 8
8 x 1 = 8
8 x 2 = 16
8 x 3 = 24
8 x 4 = 32
8 x 5 = 40
8 x 6 = 48
8 x 7 = 56
8 x 8 = 64
8 x 9 = 72
8 x 10 = 80
Multiplication table of 9
9 x 1 = 9
9 x 2 = 18
9 x 3 = 27
9 x 4 = 36
9 x 5 = 45
9 x 6 = 54
9 x 7 = 63
9 x 8 = 72
9 x 9 = 81
9 x 10 = 90
Multiplication table of 10
10 x 1 = 10
10 x 2 = 20
10 x 3 = 30
10 x 4 = 40
10 x 5 = 50
10 x 6 = 60
10 x 7 = 70
10 x 8 = 80
10 x 9 = 90
```

	10 x 10 = 100

[PT] modularização

Z - Número Primo a tempo inteiro

Faça um programa que leia dois números inteiros e determine quais os números nesse intervalo que são Primos a tempo inteiro.
Um número é Primo a tempo inteiro se for um número primo e, qualquer deslocação de um algarismo do fim para o início continua a ser um número Primo.

Exemplo:
123 : 123 (é primo), 312 (não é primo), 231 (é primo) : 123 NÃO É Primo a tempo inteiro
113 : 113 (é primo), 311 (é primo), 131 (é primo) : 113 É Primo a tempo inteiro

O programa deve ler dois inteiros, representando o intervalo de análise e visualizar em linhas separadas cada número primo a tempo inteiro do intervalo.
No final deve ser visualizada a quantidade de números primos a tempo inteiro encontrados, entre parênteses. O resultado deverá apresentar o seguinte formato:
< primo a tempo inteiro1 >
< primo a tempo inteiro2 >
...
(quantidade de primos a tempo inteiro)

[EN] modularization

Z - Full-time Prime number

Write a program that reads two integers and determines which numbers in that range are full-time primes.
A number is full-time prime if it is a prime number, and any shift of one digit from end to beginning is still a prime number.

Example:
123 : 123 (is prime), 312 (not prime), 231 (is prime) : 123 NOT Full time prime
113 : 113 (is prime), 311 (is prime), 131 (is prime) : 113 Is full-time prime

The program should read two integers, representing the parsing range, and display each full-time prime number in the range on separate lines.
At the end, the number of full-time prime numbers found should be displayed, in parentheses. The result should have the following format:
< full-time prime1 >
< full-time prime2 >
...
(number of full-time primes)

Example1:

Input	Output
10 100	11 13 17 31 37 71 73 79 97 (9)

Example2:

Input	Output
1000 5000	1193 1931 3119 3779 (4)

Example3:

Input	Output
10000 50001	11939 19391 19937 37199

	39119 (5)

Example4:

Input	Output
55443322 55443333	(0)