A line can be some maximum number of characters long
A lot of text needs to be printed neatly,
line lengths as close as possible

The compiler has a number of built in constants that are used to identify some hardware specific things. The all have an $ sign in their names, not at the beginning. Programmer defined identifiers may not contain $ signs

Example, maximum length 28.

Good:

```
The compiler has a number   |
of built in constants that  |
are used to identify some   |
hardware specific things.   |
The all have an $ sign      |
in their names, not at      |
the beginning. Programmer   |
defined identifiers may not |
contain $ signs             |
```

Bad:

```
The compiler
has a number of built in
constants that are
used to identify some
hardware specific things.
The all have an
$ sign in their names, not
at the beginning.
Programmer defined
identifiers may
not contain $ signs
```

```
% a.out 28 The compiler has a number of built in constants that are used to identify some hardware specific things. The all have an \$
sign in their names, not at the beginning. Programmer defined identifiers may not contain \$ signs
least possible cost is 113:
 113 114 117 122 129 138 145 150 157 166 177 190 205 192 193 196 201 208 217  65  66  69  74  81  90 101 104 105 113 The 3
 141 142 145 150 157 166 177 190 205 192 193 196 201 208 217  65  66  69  74  81  90 101 104 105 108 113 120 129 141 compiler 9
 192 193 196 201 208 217  65  66  69  74  81  90 101 104 105 108 113 120 129 134 141 150 155 156 159 164 171 180 192 has 4
 240 241  65  66  69  74  81  90 101 104 105 108 113 120 129 134 141 150 155 156 159 164 171 180 185 192 201 212 240 a 2
  65  66  69  74  81  90 101 104 105 108 113 120 129 134 141 150 155 156 159 164 171 180 185 192 201 212 225 240  65 number 7
 104 105 108 113 120 129 134 141 150 155 156 159 164 171 180 185 192 201 212 225 240 257  65  66  69  74  81  90 104 of 3x2 + 9
 125 126 129 134 141 150 155 156 159 164 171 180 185 192 201 212 225 240 257  65  66  69  74  81  90 101 104 109 125 built 6
 155 156 159 164 171 180 185 192 201 212 225 240 257  65  66  69  74  81  90 101 104 109 116 125 136 149 156 151 155 in 3
 176 177 180 185 192 201 212 225 240 257  65  66  69  74  81  90 101 104 109 116 125 136 149 156 151 152 155 160 176 constants 10
  65  66  69  74  81  90 101 104 109 116 125 136 149 156 151 152 155 160 167 176 187 194 203 214 227 242  56  57  65 that 5
 100 101 104 109 116 125 136 149 156 151 152 155 160 167 176 187 194 203 214 227 242  56  57  60  65  72  81  92 100 are 4x3 + 4
 140 141 144 149 156 151 152 155 160 167 176 187 194 203 214 227 242  56  57  60  65  72  81  92  95 100 107 116 140 used 5
 151 152 155 160 167 176 187 194 203 214 227 242  56  57  60  65  72  81  92  95 100 107 116 127 140 155 142 143 151 to 3
 178 179 182 187 194 203 214 227 242  56  57  60  65  72  81  92  95 100 107 116 127 140 155 142 143 146 151 158 178 identify 9
  56  57  60  65  72  81  92  95 100 107 116 127 140 155 142 143 146 151 158 167 178 191 206  20  21  24  29  36  56 some 5
  91  92  95 100 107 116 127 140 155 142 143 146 151 158 167 178 191 206  20  21  24  29  36  45  56  69  82  83  91 hardware 9x8 + 9
 142 143 146 151 158 167 178 191 206  20  21  24  29  36  45  56  69  82  83  86  91  98 103 106 111 118 127 130 142 specific 9
  20  21  24  29  36  45  56  69  82  83  86  91  98 103 106 111 118 127 130 135 142 151 160 163   5   6   9  12  20 things. 8
  82  83  86  91  98 103 106 111 118 127 130 135 142 151 160 163   5   6   9  12  15  20  27  36  47  50  55  62  82 the 4x3 + 9
 102 103 106 111 118 127 130 135 142 151 160 163   5   6   9  12  15  20  27  36  47  50  55  62  71  82  91 100 102 all 4
 126 127 130 135 142 151 160 163   5   6   9  12  15  20  27  36  47  50  55  62  71  82  91 100 101 102 105 110 126 have 5
 159 160 163   5   6   9  12  15  20  27  36  47  50  55  62  71  82  91 100 101 102 105 110 117 126 137 150 151 159 an 3
   5   6   9  12  15  20  27  36  47  50  55  62  71  82  91 100 101 102 105 110 117 126 137 150 151 154 159   1   5 $ 2
  11  12  15  20  27  36  47  50  55  62  71  82  91 100 101 102 105 110 117 126 137 150 151 154 159   1   2   5  11 sign 5
  46  47  50  55  62  71  82  91 100 101 102 105 110 117 126 137 150 151 154 159   1   2   5  10  11  14  19  26  46 in 3x2 + 46
  75  76  79  84  91 100 101 102 105 110 117 126 137 150 151 154 159   1   2   5  10  11  14  19  26  35  46  59  75 their 6
 101 102 105 110 117 126 137 150 151 154 159   1   2   5  10  11  14  19  26  35  46  59  66  75  86  99 114 131 101 names, 7
 150 151 154 159   1   2   5  10  11  14  19  26  35  46  59  66  75  86  99 114 131 150 101 102 105 110 117 126 150 not 4
   1   2   5  10  11  14  19  26  35  46  59  66  75  86  99 114 131 150 101 102 105 110 117 126 137 150 165 182   1 at 3
  10  11  14  19  26  35  46  59  66  75  86  99 114 131 150 101 102 105 110 117 126 137 150 165 182 201   1   2  10 the 4x3 + 36
  50  51  54  59  66  75  86  99 114 131 150 101 102 105 110 117 126 137 150 165 182 201   1   2   5  10  17  26  50 beginning. 11
 101 102 105 110 117 126 137 150 165 182 201   1   2   5  10  17  26  37  50   1   2   5  10  17  26  37  50  65 101 programmer 11
   1   2   5  10  17  26  37  50   1   2   5  10  17  26  37  50  65  82 101 122   0   1   4   9   0   1   4   9   1 defined 8x7 + 9
   1   2   5  10  17  26  37  50  65  82 101 122   0   1   4   9   0   1   4   9   0   1   4   9  16  25  36  49   1 identifiers 12
   0   1   4   9   0   1   4   9   0   1   4   9  16  25  36  49   0   1   0   1   4   9  16  25   0   0   0   0   0 may 4
   0   1   4   9   0   1   4   9  16  25  36  49   0   1   0   1   4   9  16  25   0   0   0   0   0   0   0   0   0 not 4
   0   1   4   9  16  25  36  49   0   1   0   1   4   9  16  25   0   0   0   0   0   0   0   0   0   0   0   0   0 contain 8x7 + 1
   0   1   0   1   4   9  16  25   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 $ 2
   0   1   4   9  16  25   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 signs 6
The compiler has a number     (9)
of built in constants that    (4)
are used to identify some     (9)
hardware specific things.     (9)
The all have an $ sign        (36)
in their names, not at        (36)
the beginning. Programmer     (9)
defined identifiers may not   (1)
contain $ signs               (0)
total 113
```

```
#include <iostream>
#include <iomanip>
#include <cstring>

using namespace std;

int main(int argc, char * argv[])
{ if (argc < 3)
  { cerr << "need line length and list of words\n";
    exit(1); }
  int linelen = atol(argv[1]);
  int numwords = argc - 2;
  char * * wordlist = argv + 2;

  int * lenword = new int[numwords];
  for (int i = 0; i < numwords; i += 1)
  { lenword[i] = strlen(wordlist[i]);
    if (lenword[i] > linelen)
    { cerr << "Impossible, " << wordlist[i] << " is too long\n";
      exit(1); } }

  int * * table = new int * [numwords];
  for (int i = 0; i < numwords; i += 1)
    table[i] = new int[linelen + 1];

  // table[word index under consideration][space left on current line]
  //    = minimum cost of the whole thing started here.
  // costs squared otherwise distribution of spaces won't matter

  int lastwordlen = lenword[numwords - 1];
  table[numwords - 1][linelen] = 0;
  for (int left = 0; left < lastwordlen + 1; left += 1)
    table[numwords - 1][left] = left * left;
  for (int left = lastwordlen + 1; left <= linelen; left += 1)
    table[numwords - 1][left] = 0;

  for (int word = numwords - 2; word >= 0; word -= 1)
    for (int left = linelen; left >= 0; left -= 1)
    { int poss1, poss2;
      if (left == linelen)
        poss1 = table[word + 1][linelen - lenword[word]];
      else if (left >= lenword[word] + 1)
        poss1 = table[word + 1][left - lenword[word] - 1];
      else
        poss1 = 0x7FFFFFFF;
      poss2 = left * left + table[word][linelen];
      table[word][left] = min(poss1, poss2); }

  cout << "least possible cost is " << table[0][linelen] << ":\n";
  for (int word = 0; word < numwords; word += 1)
  { for (int left = 0; left <= linelen; left += 1)
      cout << setw(4) << table[word][left];
    cout << "\n"; }

  int word = 0, left = linelen, total = 0;
  while (word < numwords)
  { int size = lenword[word];
    if (left != linelen)
      size += 1;
    if (size > left ||
        word + 1 < numwords && table[word + 1][left - size] != table[word][left])
    { cout << setw(left) << "" << " (" << left * left << ")\n";
      total += left * left;
      left = linelen; }
    else
    { if (left != linelen)
      { cout << " ";
        left -= 1; }
      cout << wordlist[word];
      left -= lenword[word];
      word += 1; } }
  if (left != linelen)
    cout << setw(left) << "" << " (0)\n";
  cout << "total " << total << "\n"; }
```