LAB Assignment3
Professor: Mohan Kumar
Abhishek Sharma

Summary:
Pretty_Printing problem:
It is problem of distributing the words evenly as much as possible with given width
Length or maximum allowed words in a line so that the overall paragraph look neatly.
The document managing software like MS-Word, LaTex has this kind of algorithm in their software.

Algorithm:
As in the given problem 6.6 in the book the algorithm didn't include ',' , '?', '!' at end of any word while calculating the Line break. It calculate the cost for each line in terms of square of the number of extra spaces left after filling the line with words. The total sum will the sum of all the cost for each line. For example if there are 4 line in a paragraph and each has the extra space as 2, 1 ,3 ,1 . total space = 7 hence the total cost will be = 2*2+1*1+3*3+ 1*1 = 15
Again say for same paragraph the words are arranged in such a order that the line has the extra spaces as 2, 1,2,2 so the total space will be the same = 7 and the total cost = 2*2+1*1+2*2+2*2 = 13
Hence the second arrangement will be chosen as it has less cost than the previous one.

There could be two approach for the same problem.
1) Greedy approach
2) Dynamic programming

In Greedy approach we goes on filling words in sequential order in each line without bothering about the total cost. But in this approach there is guarantied optimal total space count but not the total cost as the problem we have discussed in the previous example.
For having the optimum total cost we use dynamic programming where we compare the total cost from all the previous cost and calculate the optimum cost for the present line.
In this program the paragraph is been taken and converted into array of words and another array holds the length of each words and this array used to compute the actual line break for each line
This algorithm first compute all the possible slack cost (i.e square of extra space at end of each line) of putting words i through j in a single line for each combination of words i and j (i<j) in a matrix 'slack_matrix'. Once the slack cost has been computed we can calculate the total cost by using the recursive expression below:

```
Cost[i] = min(1 <=i <=j){ (Cost [j-1] + slack_matrix[j,i])} if i>0
                              Else =0 for i = 0;
```

Here we uses the concept of dynamic programming as we have the overlapping subproblem where the present calculation will be using the previously calculated results.

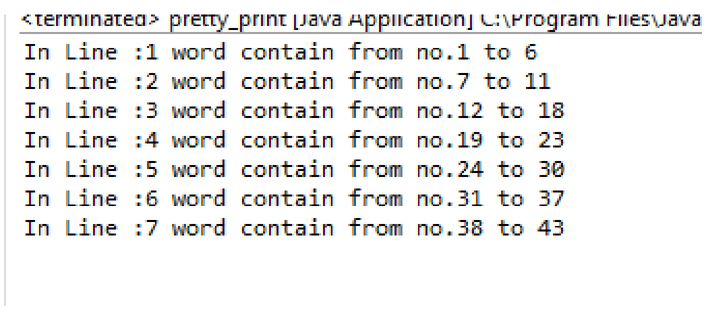
Tables: The algorithm uses two tables:
1) Slack_matix: for slack calulation
2) Cost_matrix: optimal total cost for printing words through 1 to 'i'

Data Structure Used:
In the program two data structure has been used:
Vector array: for storing the words of paragraphs at run time because the size of paragraph is unknown and vector can increase dynamically.
2D- Arrays: for storing the slack_matrix and cost_matrix for i to j words (i: for rows, J:column).

NOTE: the program only ignores few general special characters at the end of any words and not from the beginning or middle.

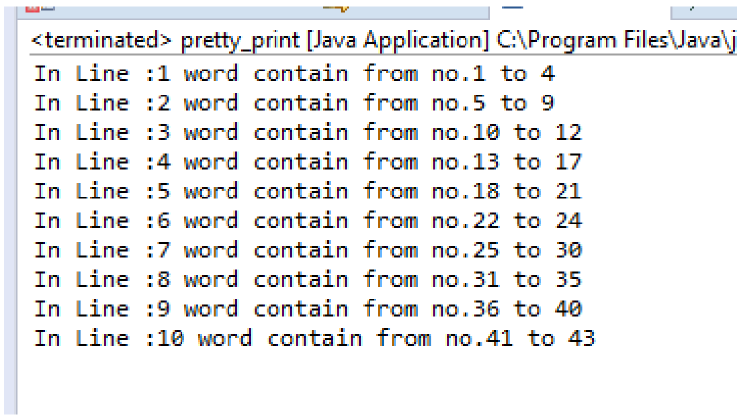
Sample Results:
For paragraph:
Call me Ishmael. Some years ago, never
mind how long precisely, having little
or no money in my purse, and nothing
particular to interest me on shore, I
thought I would sail about a little
and see the watery part of the world.

O/P:
line width of 35



```
<terminated> pretty_print [Java Application] C:\Program Files\Java
In Line :1 word contain from no.1 to 6
In Line :2 word contain from no.7 to 11
In Line :3 word contain from no.12 to 18
In Line :4 word contain from no.19 to 23
In Line :5 word contain from no.24 to 30
In Line :6 word contain from no.31 to 37
In Line :7 word contain from no.38 to 43
```

line width of 25

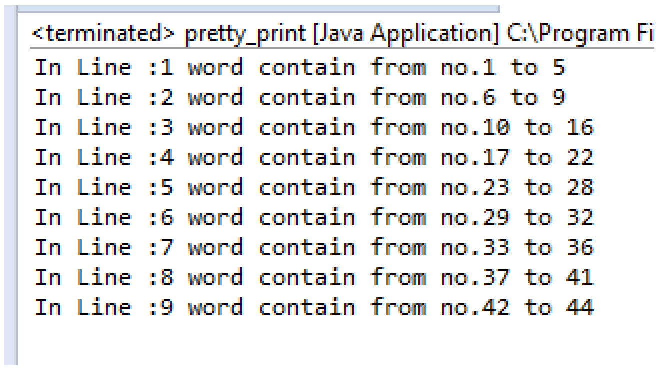


```
<terminated> pretty_print [Java Application] C:\Program Files\Java\j
In Line :1 word contain from no.1 to 4
In Line :2 word contain from no.5 to 9
In Line :3 word contain from no.10 to 12
In Line :4 word contain from no.13 to 17
In Line :5 word contain from no.18 to 21
In Line :6 word contain from no.22 to 24
In Line :7 word contain from no.25 to 30
In Line :8 word contain from no.31 to 35
In Line :9 word contain from no.36 to 40
In Line :10 word contain from no.41 to 43
```

For paragraph:

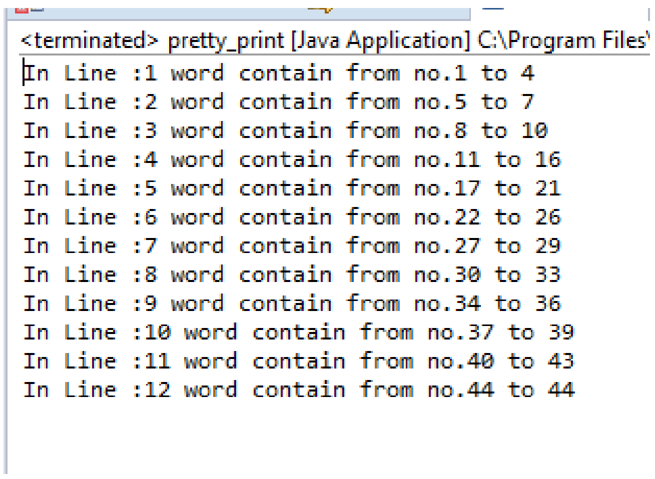
This is pretty printing algorithm.
problem doesn't include special
character at the end of any words
and execute on slack cost and
total cost matrix. it uses dynamic
programming concept and has recursive
expression which utilises the previously
computed result into the present execution.

O/P:
line width of 35



```
<terminated> pretty_print [Java Application] C:\Program Fi
In Line :1 word contain from no.1 to 5
In Line :2 word contain from no.6 to 9
In Line :3 word contain from no.10 to 16
In Line :4 word contain from no.17 to 22
In Line :5 word contain from no.23 to 28
In Line :6 word contain from no.29 to 32
In Line :7 word contain from no.33 to 36
In Line :8 word contain from no.37 to 41
In Line :9 word contain from no.42 to 44
```

line width of 25



```
<terminated> pretty_print [Java Application] C:\Program Files\
In Line :1 word contain from no.1 to 4
In Line :2 word contain from no.5 to 7
In Line :3 word contain from no.8 to 10
In Line :4 word contain from no.11 to 16
In Line :5 word contain from no.17 to 21
In Line :6 word contain from no.22 to 26
In Line :7 word contain from no.27 to 29
In Line :8 word contain from no.30 to 33
In Line :9 word contain from no.34 to 36
In Line :10 word contain from no.37 to 39
In Line :11 word contain from no.40 to 43
In Line :12 word contain from no.44 to 44
```