Algorithm for the given problem

Step 1: **S** ← Input Sentences

Step 2: **T** ← Input Terms

Step 3: **W** ← **mapSentencesAndTermsToWeights(S,T)**

Step 4: **R** ← **processStringsFromWeights(W,[ ],S,[ ])**

Step 5: display(**R**)

Procedure **mapSentencesAndTermsToWeights(sentences, terms)**

    weights Declare as array

    i ← 0

    for each **sentence** in **sentences** do

        termWeights Declare as array

        for each **term** in **terms** do

            if **term** is present in **sentence**

                append **term** to **termWeights**

            end if

        end for

        sentenceIndex ← **i**

        sumOfWeights ← number of elements in **weights**

        terms ← **termWeights**

        append {**sentenceIndex, sumOfWeights, terms**} as dictionary to **weights**

        i ← i+1

    end for

    return **filterAndSortWeights(weights)**

Procedure **filterAndSortWeights(weights)**

 N ← number of items in weights – 1

 for **i ←** 0 to n do

  if **weights[i]** == 0

   remove **weights[i]** from **weights**

  endif

 end for

 sort(**weights**) based on weights[index][**sumOfWeights**] in descending order

 return **weights**

Procedure **processStringsFromWeights(weight,result,sentences,termsUsed)**

 if **weights** is empty

  return **result**

 endif

 i ← index of first element of sorted **weights**

 append **sentences[i]** to **result**

 append each **term** present in Sentences[i] to **termsUsed**

 i ← 0

 for each **weight** in **weights** do

  for each **term** in **terms**

   if **term** is present in **weight[i][terms]**

    remove **weights[i]** from **weights**

   endif

  end for

 end for

 i ← i + 1

 return **processStringsFromWeights(weight,result,sentences,termsUsed)**