CS2100 Computer Programming Lab
Dynamic Programming

Release Date: October 26, 2015
Due Date: November 8 , 2015

Problem 1 : Punctuating a corrupted string

You are given a string of n characters $s[1, \ldots, n]$, which you believe is a corrupted text document in which all punctuations have vanished. For example the string could look something like "itwasagoodbrightandsunnyday". Your goal is to reconstruct the string by inserting appropriate spaces. You are given access to a dictionary in the form of a boolean function:
bool dict(char* s): returns true if s is a valid word, returns false otherwise.
Use dynamic programming to determine whether the given string can be reconstructed as a sequence of valid words. You may assume that calls to the function dict are of unit cost, irrespective of the length of the string. In case the string is valid, your program must output the corresponding sequence of valid words.

In order to get started, consider maintaining an array $valid[1, \ldots, n]$. The entry $valid[i]$ is 1 if the string $s[1, \ldots, i]$ can be reconstructed as a valid sequence of words; otherwise $valid[i]$ is 0. How will you populate $valid[i + 1]$?

Input Output

Assume that you are given k strings in a file. The file has the following format: the first line contains k, the number of strings. The next k lines have the k input strings. You may assume that each string is no more than 100 characters long. For each string you need to determine if a valid reconstruction is possible. In case no valid reconstruction is possible, you must output "Invalid string". Otherwise, you must output a valid sequence of words corresponding to the string.

A sample inputfile.txt and outputfile.txt are given below.

inputfile.txt	**outputfile.txt**
4	
thisisagoodtest	this is a good test
thissscanotbebroooken	Invalid string
testmeifyoucanandprintme	test me if you can and print me
howdoyouwritesuchaprogram	how do you write such a program

For testing purposes, you can assume a suitable **dict** function. While testing we will replace the dict function using our own function. Your program must accept two files – inputfile and output file as command line arguments and store the output in the outputfile.

Problem 2 : Subset-sum

In this problem we are given a set of n integers $A[1, \ldots, n]$ and a target value t, where t is a positive integer > 0. Our goal is to determine if there exists a subset of elements of the given array that sums upto t. For example, if we are given an array $A = [1, -3, 4, 7, 9, -20]$ and a target value $t = 13$, then there exists a subset $\{-3, 7, 9\}$ which sum upto t. A straightforward solution involves generating all subsets of the given array and testing whether the desired subset exists. In this problem we will develop a dynamic programming solution for the same.

Consider maintaining a matrix M of size $n \times t$ which records the following: the entry $M[i, j]$ is 1 if the sum of j can be realized using the elements $A[1, \ldots, i]$; otherwise $M[i, j]$ is 0. How do you recursively define the entry $M[i, j]$? What are the base cases? What does the entry $M[n, t]$ signify?

In case the input contains a subset which sums up to t, your program must also print the subset that realizes the sum.

Input Output

Assume that the input is given to you in a file of the following format. The first line contains the number of test cases k. The following k lines contain, first the number of elements in the array, then elements of the array and, the target value t all separated by space. The output file should contain the subset if one exists (which sums upto the target) otherwise output a line "No subset found".

input-file.txt

```
3
8 1 7 -3 9 12 30 300 6 39
8 1 7 -3 9 12 30 300 6 304
3 200 300 200 600
```

output-file.txt

```
-3, 12, 30
7, -3, 300
No subset found
```

Your program must accept two files – inputfile and output file as command line arguments and store the output in the outputfile.

Submission Guidelines

Follow the usual submission guidelines and submit two different files containing the codes for Problem1 and Problem2. Remove all temporary files and object files and make a tar.gz and upload the tar ball to moodle.