Abgabetermin: SA in der KW 12

☐ **Gr. 1, E. Pitzer** **Name** _____ **Aufwand in h** _____

☐ **Gr. 2, F. Gruber-Leitner**

Punkte _____ **Kurzzeichen Tutor / Übungsleiter** _____ / _____

T9 (2 + 4 + 4 + 4 + 4 + 6 Punkte)

Der T9-Code (*Text on 9 keys*, siehe *http://de.wikipedia.org/wiki/Text_on_9_keys*) bildet das Leerzeichen und die 26 Buchstaben des Alphabets auf die neun Dezimalziffern 0 sowie 2 bis 9 (z. B. auf den Tasten eines Mobiltelefons) ab, und zwar gemäß folgender Tabelle:

Buchstaben	*Ziffer*
a, b, c	2
d, e, f	3
g, h, i	4
j, k, l	5
m, n, o	6
p, q, r, s	7
t, u, v	8
w, x, y, z	9

(aus *http://de.wikipedia.org/)*

Damit ist eine eindeutige Abbildung von Wörtern auf ganze Zahlen möglich (z. B. wird das Wort "kiss" auf die Zahl 5477 abgebildet).

Nutzbringend (als Eingabehilfe) verwendet wird dieser Code gerade für die Abbildung in der entgegengesetzten Richtung: Eine ganze Zahl (also eine Ziffernfolge, die z. B. über die Handy-Tastatur eingegeben wird) entspricht nun allerdings mehreren Zeichenketten, wobei aber nur wenige davon tatsächlich Wörter sind (z. B. kann die Zahl 5477 nicht nur auf das Wort "kiss" abgebildet werden, sondern sinnigerweise auch auf das Wort "lips", aber auch auf das Wort "lisp" und viele weitere, leider unsinnige Zeichenketten wie z. B. "lhqr").

Für eine Ziffernfolge der Länge n gibt es mindestens 3^n und höchstens 4^n Zeichenketten, in welche diese Ziffernfolge abgebildet werden kann, je nachdem, welche Ziffern darin vorkommen – nur sehr wenige dieser Zeichenketten sind Wörter.

a) Entwickeln Sie eine Funktion `word2number()`, die ein Wort mittels iTap/T9 auf eine Ziffernfolge abbildet. Um längere Wörter und die sich daraus ergebenden sehr großen Zahlen darstellen zu können, sollen auch die Ziffernfolgen in Form von Zeichenketten dargestellt werden (z. B. soll `word2number("kiss")` die Zeichenkette `"5477"` liefern).

b) Entwickeln Sie eine Funktion `number2strings()`, die für eine Ziffernfolge (in Form einer Zeichenkette) als Funktionsergebnis in einem passenden Behälter alle Zeichenketten liefert, die sich aus der Ziffernfolge bilden lassen. Z. B sollte `number2strings("5477")` insgesamt 144 Zeichenketten von `"jgpp"` bis `"liss"` liefern.

Die Funktion aus b) ist als Eingabehilfe noch nicht geeignet, da sie – wie schon erwähnt – auch viele unsinnige Zeichenketten liefert. Hat man allerdings ein Wörterbuch zur Verfügung, z. B. jenes, das Sie unter http://tug.ctan.org/tex-archive/systems/win32/winedt/dict/de_neu.zip herunterladen können, das über 500.000 deutsche enthält, kann man mit dessen Hilfe "die Spreu vom Weizen trennen".

c) Entwickeln Sie eine Funktion `number2words()`, die für eine Ziffernfolge nur mehr solche Zeichenketten liefert, die sich auch im Wörterbuch befinden, bei denen es sich also um "echte Wörter" handelt. Gehen Sie dabei so vor, dass Sie die Wörterbuchdatei beim Programmstart in einen geeigneten Behälter einlesen, und dass Sie dann jede Zeichenkette, die Sie erzeugen im Wörterbuch suchen: wird sie gefunden, handelt es sich um ein Wort und kann in den Ergebnisbehälter aufgenommen werden. Wie beurteilen Sie die Effizienz dieses Verfahrens?

d) Eine weitere Möglichkeit dieses Problem zu lösen, besteht darin, die Wörter im Wörterbuch primär nach ihrer Länge zu betrachten: Wählen Sie einen Behälter, der es gestattet, für alle Wortlängen *n* die Wörter dieser Länge so zu speichern, dass man in einer neuen Version Ihrer Funktion `number2wordsByLength()` aufgrund der Länge der übergebenen Ziffernfolge die Wortsuche auf Wörter mit der passenden Länge einschränken kann. Die Ergebnisse sollten gegenüber c) unverändert bleiben, aber wie beurteilen Sie die Effizienz dieses neuen Verfahrens?

e) Eine noch viel nützlichere Eingabehilfe bestünde darin, dass für eine Ziffernfolge (die noch kein vollständiges Wort darstellen muss) nicht nur jene Wörter berechnet werden, die in der Länge genau dazu passen, sondern auch all jene Wörter, die sich zukünftig (durch Verlängerung der aktuellen Ziffernfolge) daraus ergeben könnten, für welche die Ziffernfolge also ein gültiges Präfix darstellt. Entwickeln Sie eine neue Funktion `numberPrefix2word()`, die für eine Ziffernfolge alle Wörter mit passendem Präfix liefert. So soll z. B. für die Ziffernfolge `"4355"` nicht nur (wie schon bisher) das in der Länge genau passende Wort "hell" sondern auch die längeren Wörter wie "hellen" und "hello" geliefert werden. Überlegen Sie gut, ob der/die bisherige/n Behälter für das Wörterbuch auch für diese Aufgabe (noch) geeignet ist/sind und verwenden Sie ev. einen anderen. Wie beurteilen Sie die Effizienz dieses letzten Verfahrens?

f) Die Benutzerfreundlichkeit lässt sich noch weiter steigern, indem Sie die zu einem Präfix gehörende Wortmenge nach der Verwendungshäufigkeit sortieren. Erweitern Sie ihr Wörterbuch um eine Möglichkeit, die Häufigkeit der einzelnen Wörter zu speichern und initialisieren Sie das Wörterbuch einerseits aus der angegeben Quelle und andererseits, indem Sie aus einem repräsentativen Text – der mindestens 20.000 Wörter umfassen sollte und idealerweise in der gleichen Sprache wie das Wörterbuch verfasst wurde – die Worthäufigkeiten ermitteln. Entwickeln Sie dazu eine weiter Funktion `numberPrefix2sortedWords()`, das für einen Präfix dieselbe Wortmenge wie `numberPrefix2word()` ermittelt, die Wortmenge aber nach der Worthäufigkeit absteigend sortiert zurückgibt.

Ein Tipp, um Ihre Nerven (und Ihren Rechner) zu schonen:

Nutzen Sie die STL möglichst gut aus und testen Sie Ihr Programm zu Beginn eventuell mit einer selbst kleinen Wörterbuchdatei (z.B. 100 Wörter), so dass Sie Fehler leichter lokalisieren können. Verwenden Sie die Wörterbuchdatei mit den über 500.000 Einträgen erst wenn Ihr Programm "halbwegs" funktioniert.