Section #4

Tyler Conklin and Kate Rydberg (Based on handouts by previous CS106B instructors and TAs, especially those of Ashley Taylor and Shreya Shankar.)

Recursive Backtracking

1. Can Make Sum

Write a recursive function named `canMakeSum` that takes a reference to a `Vector` of ints and an int target value and returns true if it is possible to have some set of values from the `Vector` that sum to the target value. For example, the vector {`1,1,2,3,5`} and target value 9 should return true (5 + 3 + 1 = 9). However, the vector {`1,4,5,6`} and target value 8 should return false.

2. Edit Distance

Write a recursive function named `editDistance` that accepts two string parameters and returns the ″edit distance″ between the two strings as an integer. Edit distance (also called Levenshtein distance) is minimum number of ″changes″ required to get from `s1` to `s2` or vice versa. A ″change″ is a) inserting a character, b) deleting a character, or c) changing a character to a different character.

Call	Value Returned
`editDistance("driving", "diving")`	1
`editDistance("debate", "irate")`	3
`editDistance("football", "cookies")`	6

Your solution must not use any loops; it must be recursive. You may not use the string functions `find` and `rfind` because they allow you to get around using recursion. Similarly, the `replace` member is forbidden. Do not construct any data structures (no array, vector, set, map, etc.) or declare any global variables. You are allowed to define helper functions if you like.

3. Longest Common Subsequence

Write a recursive function named `longestCommonSubsequence` that returns the longest common subsequence of two strings. Recall that if a string is a subsequence of another, each of its letters occurs in the longer string in the same order, but not necessarily consecutively. For example, the following calls should return the following values:

Call	Value Returned
`longestCommonSubsequence("tyler", "kate")`	″te″
`longestCommonSubsequence("hannah", "banana")`	″anna″
`longestCommonSubsequence("she sells", "seashells")`	″sesells″
`longestCommonSubsequence("janet", "cs106b")`	″″

4. Print Squares

Write a function `printSquares` that uses backtracking to find all ways to express an integer as a sum of squares of unique positive integers. For example, the call of `printSquares(200);` should produce:

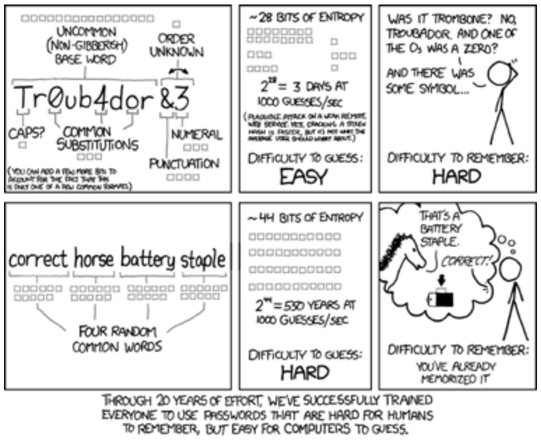
```
1^2 + 2^2 + 3^2 + 4^2 + 5^2 + 8^2 + 9^2
1^2 + 2^2 + 3^2 + 4^2 + 7^2 + 11^2
1^2 + 2^2 + 5^2 + 7^2 + 11^2
1^2 + 3^2 + 4^2 + 5^2 + 6^2 + 7^2 + 8^2
1^2 + 3^2 + 4^2 + 5^2 + 7^2 + 10^2
2^2 + 4^2 + 6^2 + 12^2
2^2 + 14^2
3^2 + 5^2 + 6^2 + 7^2 + 9^2
6^2 + 8^2 + 10^2
```

Some numbers cannot be represented as a sum of squares, in which case your function should produce no output. The sum has to be formed with unique integers. Throw an error if the integer passed to your function is negative. Assume there exists a function named display that accepts a collection of integers (i.e., Vector, Set, Stack, Queue, etc.) and prints the collection's elements in the above format.

5. Crack

Write a function `crack` that takes in the maximum length a site allows for a user's password and tries to find the password into an account by using recursive backtracking to attempt all possible passwords up to that length (inclusive).

Assume you have access to the function `bool login(string password)` that returns true if a password is correct. You can also assume that the passwords are entirely alphabetic and case- sensitive. You should return the correct password you find, or the empty string if you cannot find the password. You should return the empty string if the maximum length passed is 0 or throw an integer exception if the length is negative.


UNCOMMON (NON-GIBBERISH) BASE WORD
ORDER UNKNOWN
Tr0ub4dor &3
CAPS?
COMMON SUBSTITUTIONS
NUMERAL
PUNCTUATION
(YOU CAN ADD A FEW MORE BITS TO ACCOUNT FOR THE FACT THAT THIS IS ONLY ONE OF A FEW COMMON FORMATS)
~28 BITS OF ENTROPY
2^28 = 3 DAYS AT 1000 GUESSES/SEC
(PLAUSIBLE ATTACK ON A WEAK REMOTE WEB SERVICE. YES, CRACKING A STOLEN HASH IS FASTER, BUT IT'S NOT WHAT THE AVERAGE USER SHOULD WORRY ABOUT.)
DIFFICULTY TO GUESS: EASY
WAS IT TROMBONE? NO, TROUBADOR. AND ONE OF THE Os WAS A ZERO?
AND THERE WAS SOME SYMBOL...
DIFFICULTY TO REMEMBER: HARD
correct horse battery staple
FOUR RANDOM COMMON WORDS
~44 BITS OF ENTROPY
2^44 = 550 YEARS AT 1000 GUESSES/SEC
DIFFICULTY TO GUESS: HARD
THAT'S A BATTERY STAPLE.
CORRECT!
DIFFICULTY TO REMEMBER: YOU'VE ALREADY MEMORIZED IT
THROUGH 20 YEARS OF EFFORT, WE'VE SUCCESSFULLY TRAINED EVERYONE TO USE PASSWORDS THAT ARE HARD FOR HUMANS TO REMEMBER, BUT EASY FOR COMPUTERS TO GUESS.

Security note: The ease with which computers can brute-force passwords is the reason why login systems usually permit only a certain number of login attempts at a time before timing out. Its also why long passwords that contain a variety of different characters are better! Try experimenting with how long it takes to crack longer and more complex passwords. See the comic from https://xkcd.com/936/ above!