Assignment #3

Computer Science I – COP 3502

Objectives

1. To learn how to use recursion in programming
2. To learn how to implement known algorithms
3. To learn how to use dynamic memory allocation

Problem: Jumble Puzzle Solver (jumble.c)

Jumble is a world game in which players are given a scrambled set of letters. Players then try to make the most valid words from those letters.

For example, given the letters TCA the user could form the words CAT, and ACT.

Our program will solve jumble puzzles for players. First, the program must read in a dictionary of valid words. This file will contain English words in alphabetical order.

Then, the program must ask the user if they have a jumble puzzle to solve. If so, prompt the user for a jumbled word. Use the recursive permutation algorithm to create all permutations of the jumbled letters and then use a binary search to search for each permutation in the dictionary.

The program should output all permutations of the jumbled word that appear in the dictionary. If no permutations form a valid word, simply display a message to that effect.

When this process is completed the user should be allowed to enter additional words.

Specifications

These specifications must be followed exactly.

Input: User

1. The user will enter y or Y to indicate they have another puzzle to solve
2. The user will enter n or N to indicate they do not have any more puzzles to solve
3. The user will enter their jumbled letters in UPPERCASE
4. The maximum number of letters for a jumble puzzle is 19 letters

Input: File

1. The first line of the input file will be an integer n, that indicates how many words are in the dictionary
    a. You must use a 2D array to store the dictionary
    b. You must use dynamic memory allocation to set the size of the dictionary array
    c. The maximum number of letters for a word in the dictionary is 19 letters
2. The following n lines will each contain one valid word for the dictionary
    a. The dictionary words will be in UPPERCASE
    b. The words will be in alphabetical order

Output:

1. You must output to two different locations
    a. The screen
    b. An output file called "jumbleout.txt"
2. For each valid permutation print the following
    a. A permutation of X that is a valid word is Y.
        i. where X is the jumbled letters and Y is the permutation
3. If a jumble has no valid words print the following
    a. Sorry, no permutations of X form a word.
        i. where X is the jumbled letters

Program

1. You must implement the following algorithms discussed in class. You must write separate functions to implement these algorithms.
    a. Binary Search
    b. Recursive Permutation
        i. It is accepted that the recursive permutation only produced permutations that are the same length as the number of jumbled letters.

SampleOutput: Screen

Below is a sample output of running the program. **Note that this sample is NOT a comprehensive test.** You should test your program with different data than is shown here based on the specifications given above.

In the sample run below, for clarity and ease of reading, the user input is given in *italics* while the program output is in **bold**. (Note: When you actually run your program no bold or italics should appear at all. These are simply used in this description for clarity's sake.)

Welcome to the Jumble Puzzle Solver!

Would you like to enter a jumbled word?
Y
What word would you like scored?
TCA
A permutation of TCA that is a valid word is ACT.
A permutation of TCA that is a valid word is CAT.

Would you like to enter a jumbled word?
y
What word would you like scored?
ZYFRPQ
Sorry, no permutations of ZYFRPQ form a word.

Would you like to enter a jumbled word?
n

Sample Output: File

A permutation of TCA that is a valid word is ACT.
A permutation of TCA that is a valid word is CAT.
Sorry, no permutations of ZYFRPQ form a word.

Deliverables

One source file – *jumble.c* – is to be submitted over WebCourses.

Restrictions

Your program must be able to be compiled and executed in a standard C Development Environment. Yourprogram should include a header comment with the following information: your name, course number, section number, assignment title, and date. Also, make sure youinclude comments throughout your code describing the major steps in solving the problem.

Grading Details

Your programs will be graded upon the following criteria:

1) Your correctness

2) Your programming styleand use of white space. Even if you have a plan and your program works perfectly, if your programming style is poor or your use of white space is poor, you could get 10% or 15% deducted from your grade.

3) Your comments. Each function must have appropriate pre and post conditions. Internal comments should be used liberally – both in main and in any functions.

4) Compatibility – You must submit C source files that can be compiled and executed in a standard C Development Environment. If your program does not compile, you will get a sizable deduction from your grade.