# Computer Science 1 - Program 2
# *Scrabble*
# Assigned: 1/29/04
# Due: 2/13/04 (Friday) at 11:55pm (WebCT time)

**Objective**

1. To write a recursive function for generating permutations.
2. To execute a binary search.

**The Problem**

You are a poor Scrabble player and would like to create a computer program that you can use to "cheat" while playing. Your program will be given a set of tiles (to simplify the problem we won't allow blanks.) From that set, you are to determine which word (without regard to bonus scores such as double letter scores) will give you the maximal score. If more than one word does, you may output any of the valid choices. In order to solve the problem, you should adapt the algorithm for generating permutations presented in the text. We will provide a sorted dictionary for you to use to verify whether or not a string is a valid word.

In your implementation please adhere to the following guidelines:

1) Use an dynamic array of strings to store the dictionary.
2) Use a binary search when searching to see if a particular string is a valid word.
3) Use an recursive function to generate all permutations of the given tiles.

**References**

Textbook: Chapter 4, 5 (especially 5.2)

**Scrabble Point Values**

A = 1, B = 3, C = 3, D = 2, E = 1, F = 4, G = 2, H = 4, I = 1
J = 8, K = 5, L = 1, M = 3, N = 1, O = 1, P = 3, Q = 10, R = 1,
S = 1, T = 1, U = 1, V = 4, W = 4, X = 8, Y = 4, Z = 10

**Input File Format**

The first line of the input dictionary file will contain a single number indicating the number of words in the file. Following this, each line will contain one word in all uppercase letters. The words will appear in alphabetical order.

Here is a sample file, smalldictionary.txt

```
5
AARDVARK
ALONE
BREAK
LAKE
```

RAKING

**User Input Specification**

1. A valid string for the filename will be entered first. However, this may not specify a valid file. In this situation, exit the program after printing a brief error message.
2. The second input is guaranteed to be a positive integer (no error checking necessary) signifying the number of tile configurations to check.
3. The rest of the inputs will each be 7-letter strings of all capital letters, each standing for a set of valid scrabble tiles without blanks. (No need to check if the distribution of letters is valid.) It is possible for your tile configuration to contain a repeated letter. In this situation, the word you form can contain that particular letter as many times as tiles you have with that letter.

**Output Specification**

You should output one line for each set of 7 tiles given with the following format if a word is possible:

```
The best word you can make is XXXX which earns Y points.
```

where XXXX is the word that earns the most points from the given tiles, and Y is the number of points that word earns. Note that the best scoring word may not use all seven tiles, or may not be the longest of all possible words.

If NO word is possible, then output the following statement:

```
You can not form any word with your tiles.
```

If the dictionary file entered is invalid, output the following statement before exiting:

```
Sorry, that is an invalid file.
```

**Output Samples**

Here are two sample outputs of running the program. Note that this is NOT a comprehensive test. You should test your program with different data than is shown here based on the specifications given. The user input is given in *italics* while the program output is in bold. (Note: The following output is based upon the sample input file shown above.)

**Output Sample #1**

**Enter the name of the dictionary file.**
*smalldictionary.txt*
**How many tile configurations do you have?**
*2*
**What is tile configuration 1?**
*ABCDEFG*
**You can not form any word with your tiles.**
**What is tile configuration 2?**
*GINKRBA*

```
The best word you can make is RAKING which earns 11 points.
```

**Output Sample #2**

```
Enter the name of the dictionary file.
dictonary.com
Sorry, that is an invalid file.
```

**Grading Details**

Your program will be graded upon the following criteria:

1) Adhering to the implementation specifications listed on the first page.
2) Your algorithmic design (for generating permutations).
2) Correctness.
3) The frequency and utility of the comments in the code, as well as the use of white space for easy readability. (We're not kidding here. If your code is poorly commented and spaced and works perfectly, you could earn as low as 80-85% on it.)
4) Compatibility to either cygwin in Windows or gcc under olympus. (If your program does not compile in either of these environments, you will get a sizable deduction from your grade.)

Note: You will not be graded on the efficiency of trying out substrings for matches in the dictionary, only the proper use of the original generating permutations algorithm shown in the text.

**Restrictions**

Name the file you create and turn in *scrabble.c*. Although you may use other compilers, your program must compile and run using gcc or cygwin. Your program should include a header comment with the following information: your name, course number, section number, assignment title, and date. You should also include comments throughout your code, when appropriate.

**Deliverables**

A single source file named *scrabble.c* turned in through WebCT.