APS 105 — Computer Fundamentals
Lab #7: The Mastermind Assistant (Recursion)
Fall 2012

Important note: The course material necessary for completing this lab will be found in your lecture notes and in the Carter text up to the end of Chapter 8.

This is a two-week lab. You must use the `submitaps105f` command to electronically submit your solution by 11:59pm on Saturday, November 17, 2012.

Objective

In Lab #6, you wrote a program that simulated the game "Mastermind". In this lab, you will write a program that assists a player with the game. Specifically, your program will compute *all* potentially successful key patterns, given a guess pattern and the feedback characters for the guess pattern. This assistant program might be especially useful as an iPhone or Android app that a player could use while actually playing Mastermind!

Mastermind Assistant

As with Lab #6, your program should first ask the user for the pattern length, `n` (an integer). Then, your program should ask the user for the guess pattern she attempted, which will have `n` characters, each of which is from `a` through `f` (as in Lab #6). Next, your program should ask the user for the number of black pegs in the feedback. Finally, your program should ask the user for the number of white pegs in the feedback. Recall that the number of black pegs indicates the number of characters in the guess that are in *exactly* the correct position. The number of white pegs indicates the number of characters in the guess pattern that appear in the key pattern, but that are in the wrong position in the guess pattern. For more details on the meaning of the black and white pegs, refer back to the handout for Lab #6.

Your program must use recursion to compute and print *all* key patterns that have the potential to be the correct key pattern. It should not print any patterns that cannot possibly be the correct key pattern, due to the number of black and white pegs in the feedback. For example, if the guess pattern is `abcd` and the number of black pegs is 3, it is certain that `bacd` is *not* a key pattern, as it has 2 characters that differ from the guess.

The list of potential key patterns must be printed in the exact order shown in the examples below. Observe that key patterns, if viewed as English-language words, are printed in alphabetical order.

Example #1:

```
Enter the pattern length: 3
Input the guess pattern: abc
Enter the number of black pegs in the feedback: 2
Enter the number of white pegs in the feedback: 0
The possible key patterns are:
aac
```

```
aba
abb
abd
abe
abf
acc
adc
aec
afc
bbc
cbc
dbc
ebc
fbc
```

Example #2:

```
Enter the pattern length: 2
Input the guess pattern: ab
Enter the number of black pegs in the feedback: 0
Enter the number of white pegs in the feedback: 2
The possible key patterns are:
ba
```

Example #3:

```
Enter the pattern length: 12
Input the guess pattern: aaaaaabbbbbb
Enter the number of black pegs in the feedback: 10
Enter the number of white pegs in the feedback: 2
The possible key patterns are:
aaaaababbbbb
aaaaabbababb
aaaaabbbabbb
aaaaabbbbabb
aaaaabbbbbab
aaaaabbbbbba
aaaabaabbbbb
aaaababababb
aaaababbbabb
aaaababbbbab
aaaababbbbba
aaabaaabbbbb
aaabaababbbb
aaabaabbbabb
aaabaabbbbab
aaabaabbbbba
aabaaaabbbbb
aabaaababbbb
```

```
aabaaabbabbb
aabaaabbbabb
aabaaabbbbab
aabaaabbbbba
abaaaaabbbbb
abaaaababbbb
abaaaabbabbb
abaaaabbbabb
abaaaabbbbab
abaaaabbbbba
baaaaaabbbbb
baaaaababbbb
baaaaabbabbb
baaaaabbbabb
baaaaabbbbab
baaaaabbbbba
```

You may assume that the user enters valid input. For example, for patterns of length 3, the sum of the number of black and white pegs can never exceed 3.

For full marks, your program must compute its result in a reasonable amount of time for large pattern lengths, such as 10.

Write your program in a file called `Lab7.c`.

Marking

This lab will be marked out of 10. A marking program will be used to automatically mark your lab. The marking program will use the last version of the lab files you submitted using the `submitaps105f` command.

Full marks are given if your program works correctly, fewer if not, and zero if it cannot be compiled. Late submissions or submissions with an incorrect filename will result in a mark of 0 for the entire lab. The deadline will be strictly enforced, so avoid last minute submissions.

Important: Any submitted solution that does not use recursion will be assigned a mark of 0.

You can run a testing program, called `tester`, yourself to test the correctness of your solution. At the command line in your ECF account, run:

```
/share/copy/aps105f/lab7/tester
```

in the same directory as your solution file. The testing program will use a number of test cases to test your solution, and report success if your solution produces output that is identical to the expected output. Some of these test cases will be used by the marking program as well, but the marking program will also be using other test cases that are not included in the testing program to test the correctness of your program. This implies that even though you do not have access to the marking program, you will obtain at least partial marks if all

of the test cases in the testing program report success with your solution.

Your mark for this lab will contribute 3% to your mark in the course.

What To Submit

When you have completed the lab, use the command

```
submitaps105f 7 Lab7.c
```

to submit your file. Make sure you name your file exactly as stated (including lower/upper case letters). Failure to do so will result in a mark of 0 being assigned. You may check the status of your submission using the command

```
submitaps105f -l 7
```

where `-l` is a hyphen followed by the letter *ell*. You can also download a copy of your submission by running the command:

```
/share/copy/aps105f/lab7/viewsubmitted
```