Recursion Lab Suggestions

isPalendrome

This function can be implemented as a “tail recursive function” — that is, it can be written as an iterative program and then converted to recursive using the loop termination as base case and the body of the loop as the recursive step.

There are two ways to approach the problem. One is to reverse the string and compare the reversed string with the original string — this does not work particularly well for a recursive approach and takes more effort and memory. The second way is to compare the first location with the last (length-1) and the second with the next to last (length-1) and so forth. If you reach a comparison that does not match, you do not have a palindrome. If you get to the middle of the string without failing to match, you have success.

Recursion Lab Suggestions

printAllSubsets

This is a harder problem. It is much simpler to design as a recursive solution than as an iterative solution and so the approach above of creating an iterative solution first is not a good approach.

The trick is to look at the progression of subsets from sets of size one on up.

For a set of size one, there are two subsets:

“A” -> “A” and “” (The set of one item and the set with none)

For a set of size two, there are four subsets:

“BA” -> “BA”, “B”,

“A”, “”

Realize that “BA” and “B” are the subsets of “A” with a “B” attached. The other two are the subsets of “A”.

For a set of size three, there are eight subsets:

“CBA” -> “CBA”, “CB”, “CA”, “C”

“BA”, “B”, “A”, “”

For a set of size four, there are sixteen subsets:

“DCBA” -> “DCBA”, “DCB”, “DCA”, “DC”, “DBA”, “DB”, “DA”, “D”

“CBA”, “CB”, “CA”, “C”, “BA”, “B”, “A”, “”

So, the pattern is that at each time, you take the subsets of the previous level and have two cases — one adds the new first character and the other adds the empty string.

When implementing this, you want to have two arguments. One for the starting string and the other for the string being created as you find subsets.

The base case is when you pass in a string of length zero and it displays the string created and returns.

The rest of the function is two calls; both call the recursive function and remove the first character from the starting string to create a new starting string. The difference between the two is as shown above in the examples (one adds the first letter of the starting string, the other does not”