Jason Anrico

CSC 212 – Data Structures

Assignment7

Make sure your code is readable and well-documented. Each function must also begin with a title block that describes the task of the function, input parameters and return value.

1. Write a recursive function that takes a string as an input and returns the reverse of the string.
2. Write a recursive function rec\_string that produces the output shown below for the corresponding function calls. Write a main function to test the function.

Method call rec\_string(‘abcde’), will produce the following output:

\*

e

de

cde

bcde

abcde

Method call rec\_string(‘abc’), will produce the following output:

\*

c

bc

abc

1. Write a recursive function for Euclid's algorithm to find the greatest common divisor (gcd) of two positive integers. gcd is the largest integer that divides evenly into both of them. For example, the gcd(102, 68) = 34. You may recall learning about the greatest common divisor when you learned to reduce fractions. For example, we can simplify 68/102 to 2/3 by dividing both numerator and denominator by 34, their gcd. Finding the gcd of huge numbers is an important problem that arises in many commercial applications. We can efficiently compute the gcd using the following property, which holds for positive integers p and q:

If p > q, the gcd of p and q is the same as the gcd of q and p % q.

1. Draw the 11-item hash table resulting from hashing the keys 12, 44, 13, 88, 23, 94, 11, 39, 20, and 16, using the hash function h(i) = (2i + 5) mod 11 and assuming collisions are handled by linear probing.
2. How many comparisons are needed to find39 in the table?

**One comparison would exist, each element is in a single slot. When the keys are hashed using linear probing, index 6 has 94 in its slot. x(39) = 6 and**

**39 !=94.**

1. How many comparisons are needed to determine that 5 is not in the table? Briefly explain.

**h(5) = 4, which means 20 is in index 4. 20 != 5 using linear probing, since one slot is occupied, one comparison is made.**

1. Draw the 11-item hash table of the previous exercise, assuming collisions are handled by chaining?
2. How many comparisons are needed to find39 in the table?

**h(39) = 6 so we are searching for index 6. Since we chained the nodes in index 6 they are 94 and 39. Therefore we need two comparisons.**

1. How many comparisons are needed to determine that 5 is not in the table? Briefly explain.

**h(5) = 4 which means index for is searched. Index 4 contains 44 and 16. Since two elements exist in index 4 we must use two comparisons**