Algorithms Exercises

# Summary

These exercises are meant to reinforce knowledge from the “Algorithms in C#” Powerpoint

# Brute force (linear search)

1. Write a function called IsFirstCharRepeated that takes in a string and returns a bool.
2. The function should return true if the first character is repeated anywhere else in the string
3. Write a unit test and test this function.

# Recursion

1. Write a function called ReverseString that takes in a string and returns a string. The function should return the string passed in, but in reverse. The function should solve the problem using iteration.
2. Write a unit test and test this function.
3. Solve the same problem using recursion and test it again.
4. Write a function called GetSumBetweenNumbers that takes in an int min and an int max and returns an int. The function should get the sum of all the numbers between (and including) min and max. The function should solve the problem using iteration.
   1. If min > max, the function should return 0
5. Write a unit test and test this function.
6. Solve the same problem using recursion and test it again.
7. Write a function called XToTheYPower that takes in an int x and an int y, and returns int. The function should return x^y. Solve this using iteration, don’t use Math.Pow()
8. Write a unit test and test this function.
9. Solve the same problem using recursion and test it again.

# Divide-and-conquer

1. Write a function called MultiplyList that takes in a List<int>. It should return the product of all numbers in the list. Solve this problem using iteration.
2. Write a unit test and test this function.
3. Solve the same problem using a recursive divide-and-conquer algorithm and test it again.

# Greedy

1. Write a function called FillRoomWithBoxes that takes in an int called roomSize, a List<int> called possibleSizes, and a List<int> called boxes.
   1. The possibleSizes list should contain all possible box sizes. You can assume this list is sorted in descending order.
2. The function should implement a greedy recursive algorithm to fill the room as full as it can get with the least number of boxes, and should store the size of each box used in the boxes list, one entry per box
   1. For example, if your possible sizes list contained 7, 3, 1, and your roomSize was 25, the boxes list should contain 7, 7, 7, 3, 1 when your function exits
3. Write a unit test and test this function