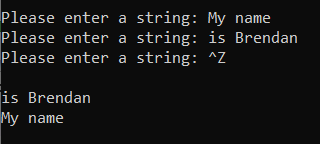
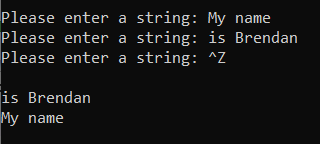
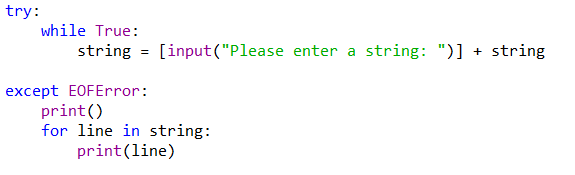
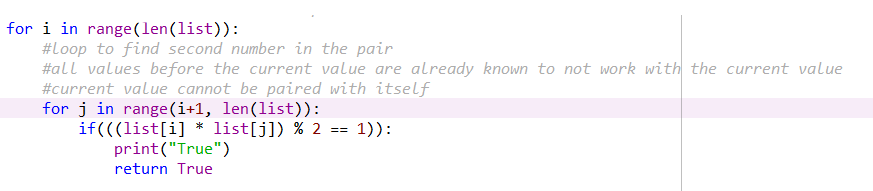
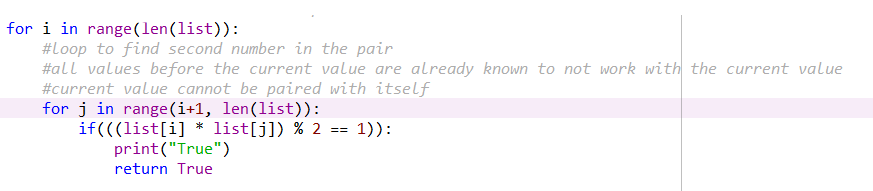
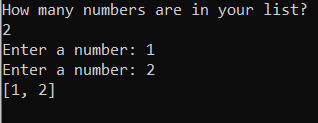
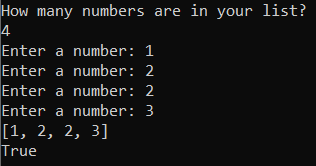
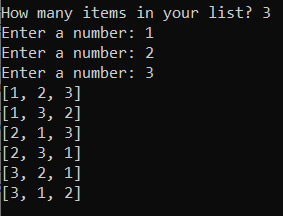
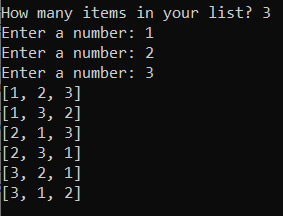
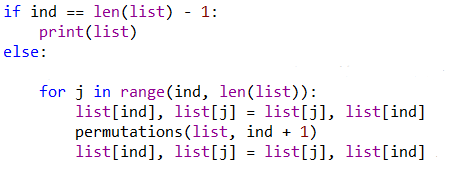
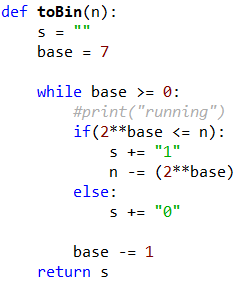
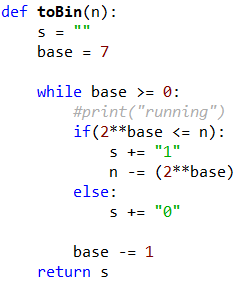
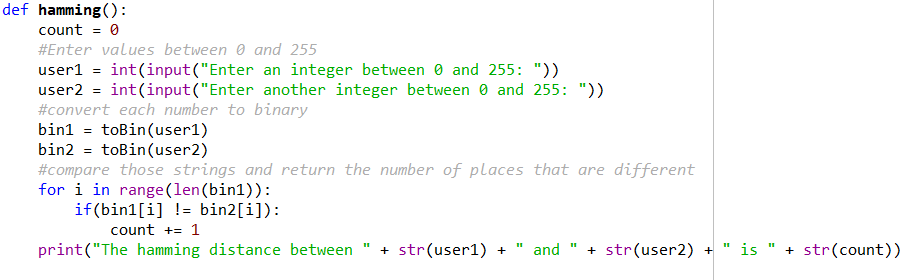
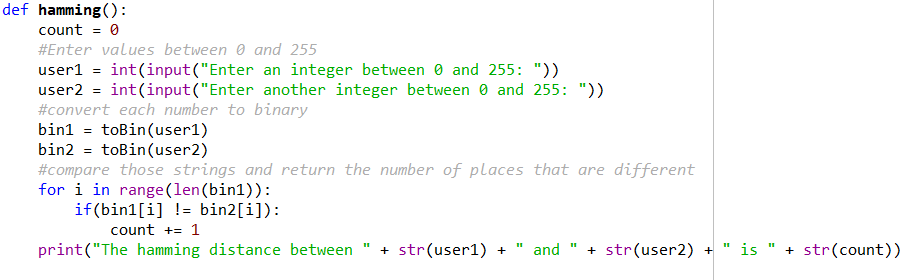
***well-documented summary that includes the annotation/justification of your algorithm and programming solution and the experimental results with the screenshots of your program testing you test. It should be at least 2 pages MS WORD or PDF file, (the more formal the better.) Note that screenshots should be readable to gain the full credits. The instructor will use the screenshots for student’s program demo.***

This document will rationalize the algorithms and functions used in my HW1 for CSC 310

1. The first portion of the assignment was relatively easy. All that needed to be done was to build a list of strings, provided by the user, then output those strings in the opposite order of how the user entered them. For example, if the user entered:  
     
   “My name”  
   “Is Brendan”  
   Ctrl+Z   
     
   then the program would output:   
     
   “Is Brendan”  
   “My name”  
   The difficult part was catching the EOFError, as I had not done that in python before. All it required was a try – except block, where the error being tested for was the EOFError, which the user could force, by inputting the key combination Ctrl+Z, in the place of any other string. For the sake of simplicity, I added any new string input by the user to the beginning of the list, so that I only had to loop through the list normally, rather than starting at the end and working towards the beginning of the list at the time of the final output.  
   
2. For the second part of the assignment, I needed a list of integers, which I prompted the user for the length and contents of. After that I needed to determine if any pair of those integers could be multiplied together to get an odd number, which was accomplished with a pair of nested loops. One to determine the first integer in the pair, another to determine the second. The inner loop started at one more than whatever value the outer loop held, since all pairings involving values earlier in the list had already been found to produce an even product.If the user defined a list of the values [1, 2, 2, 3] the program would return true. However, a list of the values [1, 2] would not.  
   
3. The third portion also required a list of integer values, and as such starts out in a similar way requesting the length and then the contents of the list. However, it prints all possible permutations of the entered list, which means that every possible ordering of the list’s contents is shown. For example:  
   This was done by taking each individual value, and placing it at the front of the list, then taking each of the remaining values and putting them in the second space in the list, and so on until no more values were available to be moved. Due to the recursive call being within the loop, this brought out all possible permutations. The variable ind was a variable used to hold whatever place values were being placed into, i.e. first, second, or third  
   
4. The final portion of the assignment needed to determine the Hamming difference between two integers. I chose to do this with the standard 8-bit model, so values ranging between 0 and 255. Hamming difference is essentially the number of places where the binary representation of two numbers differs, and so the two numbers the user entered needed to be converted to binary strings. I created a separate method to do this, which started of at a power of 7 and decremented down all the way to 0(represented by the variable base). These numbers represented the bits in the binary string. With that system set up, all that was needed was to compare 2^base to whatever integer was being converted. If the number was greater than or equal to 2^base, a 1 was added to the string. Otherwise, a 0 was added. After all 8 bases had been looped through, the resulting string was returned.   
   These returned strings were then compared to find their differences, the number of which was printed out to the user.  
   And so, values of 0 and 255 should return a difference of 8, while values of 1 and 3 should return a difference of only 1  
   