This test is out of 40 points with 5 points extra credit.

1. (10 points) Create the function *countString(target,lookfor)* which will be given two (non-empty) strings, and will count the number of times the *lookfor* string can be found inside the string *target*, case-independently. Examples:
*countString(“ABC”,”BC”) -> 1
countString(“aaaa”,”Aa”) -> 3
countString(“Hi there, Fred”, “Hi there Fred”) -> 0*
2. (10 points) Create the function *addIntegers(s):* You are given a string *s* containing a lot of items including words, integers, floating-point numbers, elephants, etc. separated by semi-colons. Return the sum of all the integers, or 0 if no integers are found. You may use built-in Python functions, or helper-functions of your own. Example:
*addIntegers(‘what?;5;4a;45.88;-2;bleep;12’) -> 15*
3. (10 points) You have read the entire contents of a file containing an English-to-Klingon dictionary into the string variable *KL*, the beginning of which looks like this:
KL = ’’’horse=Sargh
nose=ley’
python=python
sleep=yIQong
house=juH qach
in=pa’
into=vaj
tin=tin
…..
…..

’’’
Of course, the string contains very many more lines (each separated from the next line by “\n”). Each line contains the English word and then the Klingon translation. There are no empty lines to cause you grief. Create the function *Eng2Kling(s)* which will be given a phrase in English (words separated by spaces) and will translate it, word by word, into Klingon. Example of the common English phrase:
*Eng2Kling(“horse nose in house”) -> “Sargh ley’ pa’ juH qach”*

1. (10 points) Create the function *Interleave(A,B)* which will be given two lists and will return a list that interleaves the elements from the two input lists by taking elements alternately from each. That is, *Interleave()* effectively zips the two lists together for as long as each list has an element available. If one list is exhausted first, include the remaining elements of the other. Either or both lists may be empty.
Examples:

Interleave([‘a’,’b’,’c’],[2,4,6]) -> [‘a’,2,’b’,4,’c’,6]
Interleave([“hi”, “there”],[1,5,-3,6,7]) -> [“hi”,1,”there”,5,-3,6,7]
Interleave([3,4,5],[‘whatever’]) -> [3,’whatever’,4,5]
Interleace([],[‘hi’,’there’]) -> [‘hi’,’there’]

1. (5 points, extra credit): Create (almost) the same function *countStringRecursive(target,lookfor)* as in Q1, but which will be a recursive function that counts the number of times the string *lookfor* appears inside the string *target*. This time, unlike question #1, the match must be case-dependent (i.e. “Fred” does not match “fred”). Examples:

*countStringRecursive (“ABC”,”BC”) -> 1
countStringRecursive(“aaaa”,”aa”) -> 3
countStringRecursive (“aaaa”,”Aa”) -> 0
countStringRecursive (“Hi there, Fred”, “Hi there Fred”) -> 0*