CS 1410: DNA Sequencing

Human Genome Project

In order to complete the Human Genome Project, geneticists were required to sequence DNA. This process determines the correct order of DNA strands, which are made from four bases: adenine `A`, guanine `G`, cytosine `C`, and thymine `T`. To do this, the ends of two incomplete DNA strands are compared to see if they match when overlapped. If so, it is possible that these DNA strands are two pieces of a larger whole. Essentially, the goal of DNA sequencing is to piece together tiny fragments of the much larger original, called a genome.

Here is an example of two DNA strands that match when overlapped:

```
Strand 1: ACGGACATAGTCATT
Strand 2:      CATAGTCATTTCATG
Combined: ACGGACATAGTCATTTCATG
```

Assignment

You will make a program to find the best match between a target DNA strand and several candidate DNA strands. Your program will read the target DNA strand from one file (containing just one line) and the candidate DNA strands from a second file (containing one line for each of the candidate DNA strands). See the examples below.

target_strand.txt:

```
TTATAGTGATATACGTGCTTAGGTAGTGCAGAGACACAACTTATAGAGTGAGGCCAGCTCACGAGCTCTAGAAGCCCAAA
```

candidate_strands.txt:

```
TATTGTGCTCTATAGCTCCAGGCACATCCCTTGACGGATTGGGGACTGTCTTGACGAAAGTTCGGAGGTAGAAAAGTCCA
GACGACACCCTGGCAAAGGTCACGTCATGGGTGGAGTACTTATACCGGCAGCAGAGCGATCTGCTACCTATCTTCATGAT
CACGAGCTCTAGAAGCCCAAACTGTGACGCAATTGCCGGGCTAAAACTATGCTAAGAAATCCCCATTACCAGAGTCTTAG
TGAGCCGTTGGGCAGTTAACGGATTTTACTCGTCGCTGCCTGAAGTGCCAAATTTACCAAAAACCGGATAACTTCATGCA
CTTATAGAGTGAGGCCAGCTCACGAGCTCTAGAAGCCCAAATTGCTACTGTGCCGCTGCGCACCGCATGATCGCAGTCAG
TTAGAGGAATTGGACGGCACTCGGACACAAGCTCACGCCCCATACTTTAGCACCGAATATCGACTAAGCATAGTTGATCT
AGCAAGAGTTGGTATCTCTAGGGGCTTCTCCGGACGCAACGACGCGTCTGACAGTTCAGGTTGTTATGACCCGGGTGTGA
CTATGGTTAGGCAACTTCCACGCTATCCCTCGACCACGGCTCGTGGAGCCGTACCGGTGTATTTTGTTGCTGCTAATATT
GTAGCACGCAGTTCGAGTCACCCGGAAGCAGCGAAACGTTCGGCAACTACAAACTCCAATCTTGTATTCGGGTGCCTTTT
CGATTGTCTGTGTTCTGCATGAGCACAATAAGTACAAGTCGAACTGGTATTTACTAAAGTCCGCATATTGTACGGTACGT
```

In order to determine which candidate strand is the best match, you should compare the target strand to each of the candidate strands. The best match is the candidate strand with the largest number of overlapping bases (the number of contiguous bases that are in common). For example, the two strands shown above have 10 overlapping bases. To make your job easier, simply compare the end of the target strand to the beginning of each candidate strand (you do not need to compare the beginning of the target strand to the ends of the candidate strands).

Instructions

For this assignment you need to download the dnaSequence.zip folder. You should write your code in file named `dnaSequencing.py`. Your file should be added to the same folder as the downloaded files.

The download includes unittest files which are designed to check your completion of the functions below. *This is the same way CodeGrinder grades drills.* You should complete each function one-at-a-time. You are welcome and encouraged to take a look at the unittest file and see how it works. It is basically several automated calls to your code and checks for expected outcome.

Your assignment is to create the following functions. The functionality for each function is described below. You must follow the specifications exactly, but may choose your own method for solving the problem described for each. Once you have completed a function you should run the unittest for that function and have it pass all tests. Fix any errors, warnings, and/or failures.

- `fileToList`
- `returnFirstString`
- `strandsAreNotEmpty`
- `strandsAreEqualLengths`
- `candidateOverlapsTarget`
- `findLargestOverlap`
- `findBestCandidate`
- `joinTwoStrands`
- `main`

`fileToList`

The function `fileToList` receives one parameter `filename` a string. It should return a list of the lines in the file. *Remove the newline character from the end of each line.* If the file is empty or does not exist return an empty list.

```
fileToList('tar_easy.txt') -> ['ABCDEFG']
fileToList('can_easy.txt') -> ['GHHHHHH', 'FGHHHHH', 'EFGHHHH', 'CDEFGHH', 'DEFGHHH']
fileToList('empty_file.txt') -> []
fileToList('file_does_not_exist.txt') -> []
```

`returnFirstString`

The function `returnFirstString` receives one parameter `strings` a list of strings and returns the first string. If the list is empty the function should return an empty string.

```
returnFirstString(['aaa', 'bbb', 'ccc']) -> 'aaa'
returnFirstString([]) -> ''
```

`strandsAreNotEmpty`

The function `strandsAreNotEmpty` receives two parameters `strand1` and `strand2`, both strings. It returns `True` if both strands have a length greater than zero, or `False` otherwise.

```
strandsAreNotEmpty('', 'aaa') -> False
strandsAreNotEmpty('aaa', 'bbb') -> True
strandsAreNotEmpty('aaa', '') -> False
```

`strandsAreEqualLengths`

The function `strandsAreEqualLengths` receives two parameters `strand1` and `strand2`, both strings. It returns `True` if the length of both strands are equal or `False` otherwise.

```
strandsAreEqualLengths('aaa', 'bbb') -> True
strandsAreEqualLengths('aa', 'bbb') -> False
strandsAreEqualLengths('aaa', 'bb') -> False
```

`candidateOverlapsTarget`

The function `cadidateOverlapsTarget` receives three parameters `target` a string, `candidate` a string, and `overlap` an integer. It checks to see if the target and candidate strands have an overlap of `overlap` characters. The function should return `True` if they overlap or `False` otherwise.

```
target = 'ABBBBBA'
candidate = 'BABBBAA'
candidateOverlapsTarget(target, candidate, 1) -> False
candidateOverlapsTarget(target, candidate, 2) -> True
candidateOverlapsTarget(target, candidate, 3) -> False
candidateOverlapsTarget(target, candidate, 4) -> False
candidateOverlapsTarget(target, candidate, 5) -> False
candidateOverlapsTarget(target, candidate, 6) -> False
candidateOverlapsTarget(target, candidate, 7) -> False
```

`findLargestOverlap`

The function `findLargestOverlap` receives two parameters `target` and `candidate`, both strings. It should find the largest overlap and return the size of the overlap. If either strand is empty or the strands are not the same length, return `-1`. Use the functions you have already written, strandsAreNotEmpty and strandsAreEqualLengths.

```
findLargestOverlap('abcd','cdef') -> 2
findLargestOverlap('TAGGAG', 'GGTAGA') -> 1
findLargestOverlap('aaaa', 'bbbb') -> 0
findLargestOverlap('', 'hijk') -> -1
findLargestOverlap('abc', 'abcd') -> -1
```

`findBestCandidate`

The function `findBestCandidate` receives two parameters `target` a string, and `candidates` a list of strings. It examines each candidate in the candidates list and determines the candidate with the largest overlap. You can use the function findLargestOverlap to do this. If two candidates have the same overlap keep the first one. The function returns a tuple containing the candidate string with the largest overlap and the overlap. If no candidates overlap you should return an empty string for the candidate and 0 for the overlap.

```
findBestCandidate('ABC', ['BBC', 'BCC', 'BCA', 'CBA']) -> ('BCC', 2)
findBestCandidate('AAA', ['BBB', 'CCC']) -> ('', 0)
findBestCandidate('ABCD', ['ABCC', 'CDCD', 'BCDE']) -> ('BCDE', 3)
```

`joinTwoStrands`

The function `joinTwoStrands` receives three parameters `target` a string, `candidate` a string, and `overlap` an integer. It joins the target and candidate strands together merging them and returns the joined strand.

```
joinTwoStrands('abcef', 'cefgh', 3) -> 'abcefgh'
joinTwoStrands('TAGAGGT', 'AGGTTTG', 4) -> 'TAGAGGTTTG'
joinTwoStrands('TAGAGGT', '', 0) -> 'TAGAGGT'
```

`main`

The function `main` receives no parameters and returns nothing. The function should start by asking the user for the filename of the target strand file and candidate strands file (hint: use functions; fileToList, returnFirstString). After determining which of the candidate DNA strands is the best match (hint: use findBestCandidate), print the combined strand (hint: use joinTwoStrands).

```
python dnaSequencing.py
Target strand filename: tar1.txt
Candidate strands filename: can1.txt
GTCGCGTTCAGGCGCATTAAGTTAGTCGGAG
```

Finishing Up

Lastly add this snippet at the bottom of your file which will execute your `main()` function when you run dnaSequencing.py but will allow it to be imported into the unittest files without executing the main function.

```
if __name__ == '__main__':
    main()
```

Pass-off instructions

1. To pass off this assignment you need to show your completed program to the lab assistants.
    - Show them your `dnaSequencing.py` code
    - Run `test_all.py` - **All tests MUST pass!**
    - Run `dnaSequencing.py`
    - *The lab assistant may additional tests they want you to run*
2. Upload your dnaSequencing.py file to canvas, please add a comment to the top of the file with your

name and time your class meets.