Introduction to Bioinformatics – AS 250.265
Assignment 1

This assignment will test your ability to apply basic programming and computer knowledge to molecular biology problems. It is due February 10 at the start of class. You are expected to submit the programming parts of the assignment through the course website. Please submit each of your programming solutions as a separate file, for example, `question-01a.py` or `q01a.py` as the solution for question number 1a. Additionally, please make sure your name, email address, and the question number are listed as comments at the top of each file you submit. If you like, a template is available with the assignment material on the website that includes these fields for you to fill out. The written parts of the assignment may be submitted through the website as well, or they may be submitted on paper at the start of class. For each written question, try to limit your response to two or three sentences. This assignment is worth 100 points.

Question 1: Bubble Sort (30 points)

For this exercise, you will have to learn some new Python syntax for looping. In class, we discussed while loops; here, will use for loops to sort a list. An example of a for loop is below:

```
sum = 0                          # Initialize sum
for j in range(10):
    print j
    sum = sum + j                # Increment sum
print sum
```

This code fragment prints a list of numbers from 0 to 9, and then finally it prints the sum of all the numbers from 0 to 9, 45. You can download this fragment from the course website. In terms of a while loop, the above code is equivalent to the following code:

```
sum = 0                          # Initialize sum
j = 0                            # Initialize counter
while j < 10:
    print j
    sum = sum + j                # Increment sum
    j = j + 1                    # Increment counter
print sum
```

As you can see, the first way is a little more compact. How does it work? The secret is in the range function. Look this function up in the Python tutorial (Hint: it's in section 4.1). The range function takes as its argument 10 and then returns a list of 10 elements, starting at zero. Then, the for statement iterates through the loop, assigning a new value to j each time until it has gone through the entire list given by range. Thus, a simple way to traverse the elements in a list would be as follows:

```
my_list = [1, 2, -4, 3, 8]
for j in range(len(my_list)):
    print my_list[j]
```

Do you see how we took advantage of the len function to tell range how many elements to generate? Python evaluates `len(my_list)` first, which is 5 (the length of the list). Then, it uses that argument (5) in the call to range, so range returns `[0, 1, 2, 3, 4]`. These values are then assigned to j each of the five times the for statement loops.

Note that range can be used more generally. A function call of `range(4, 10)` will return the list `[4, 5, 6, 7, 8, 9]`. It starts at four a stops just shy of ten.

Now, download the bubble sort file from the assignment web page and study it carefully. It implements an algorithm called "bubble sort" that sorts a list of numbers from least to greatest. (Hint: it is called bubble sort because the smaller numbers "percolate" to the front of the list.) Now, answer the questions below.

a. [written] Examine lines 13, 14, and 15 (they are marked in the python file). Ignoring the loops and the if statement for now, what is it that these instructions are doing? (5 points)

b. [written] If lines 13-15 were to be replaced with the following lines, what would happen? Is this correct or incorrect behavior? (5 points)

```
my_list[i] = my_list[j]
my_list[j] = my_list[i]
```

c. [written] Now consider the entire program. Given that `my_list = [9, 4, 6, 7]`, simulate the code by hand. To do this, you will have to write out a table where one column is i, the next is j, and the third is the state of the list after line 15 at each step in the loop. (Hint: Your final list should be sorted, and your table with have 10 rows.) (5 points)

d. [written] Given a list of size N, exactly how many times would this algorithm loop? For example, your hand simulation above for a list of size four needed 10 iterations (or 10 rows) before the algorithm completed. Derive a formula for the general number of times the algorithm will loop before sorting a list of size N. Given that the fastest sort algorithm can sort a list with N*log(N) loops, is bubble sort good or bad for long lists? (Hint: $\sum_{i=1}^{N-1} i = \frac{N^2 - N}{2}$) (10 points)

e. [programming] Change the top of the template file to reflect your name and course information. Modify the program so that, rather than sorting from least to greatest, it sorts from greatest to least. Submit your solution to the web server. (Hint: The shortest solution will only change one line.) (5 points)

f. [programming] **Bonus Question (5 pts):** This implementation of bubble sort is slower than it has to be. Submit a program that modifies the two for loops to produce a faster version of the algorithm. This faster algorithm should avoid unnecessary comparisons.

Question 2: Protein Strings (30 points)

In the field of bioinformatics, most often proteins and DNA sequences are represented using strings. As discussed in class, Python strings are somewhat limited in that they cannot be manipulated in place as lists can. In the following example, consider the two ways of representing the sequence Arg-Ala-Leu-Met-Gly:

```
seq1 = ['R', 'A', 'L', 'M', 'G']
seq2 = 'RALMG'

# Test read access
print seq1[3]
print seq2[3]

# Test write access
seq1[3] = 'K'       # K ->Lysine
seq2[3] = 'K'       #
```

Download the above example from the assignment website and try to run it. You'll find that Python cannot process the final line, because seq2 is a string and cannot be modified once created. As you work with Python, you'll need to be able to convert between strings and lists of characters frequently.

Converting from a string to a list of characters is simple. Just use the built-in Python function `list`. This function will take a string and return a list of characters generated from the string, leaving the original string untouched.

```
str = 'MGSRQ'
lstr = list(str)
print lstr
lstr[2] = 'H'
```

This code fragment would cause python to print `['M', 'G', 'S', 'R', 'Q']`, and the final statement would be acceptable. Unfortunately, converting from a list of characters to a string is a little more complex. The function that does this, `join`, is tucked away in the string module. To access this function, you first have to *import* the module (allowing Python to reference the functions stored in that module), then, when you call the module, you must specify the module name first. Of course, there are other ways to do this (have you read the Python tutorial yet?), but for our purposes, consider the following example:

```
import string

lstr = ['M', 'G', 'S', 'R', 'Q']
print string.join(lstr, '')
```

The above code will print `'MGSRQ'`, exactly like we want. Note that the second argument to the join function is the separator text to be inserted between each list element. What happens when you only give one argument and omit the `''`? What happens when you omit the first line?

Download the string locator code from the assignment website. This program tries to find a target string in the context of a longer string. If it is successful, it prints the index in the master string where the target can be found. Otherwise, it prints -1.

a. [written] Describe in words how the algorithm works. What is the purpose of `tpos`? What is the purpose of `ptr`? What happens if multiple matches occur in the master string? (10 points)

b. [programming] Modify this program to print out all the matches of the target string in the master string, instead of just one match. Your program should print out one match index per line, or -1 if no matches are found. Matches should not be repeated in the output (no duplicate lines). Assume simple, non-overlapping targets. (5 points)

c. [programming] Now modify the program so that, instead of printing each match index on a single line, it stores the match indices in a list. Convert your code to a Python function (tutorial section 4.6) that takes as input a target and master string and returns the list of matches. The list should be empty if no match is found. (5 points)

d. [programming] Go back to the original string locator code, which can only match one occurrence of the target in the master string. Modify this code so that it can perform a generalized find and replace operation on a master string. Create a new variable, `newtext`, that will replace the target string in the master string. Then, append code after the first for loop to perform the replacement. Your program should print the location of the target in the original master string as well as the new master string after replacement. (Hint: using the `append` function, build the final string character by character, inserting the new string when you reach the location found by the first loop. Then, `join` the resulting character list.) (10 points)

Question 3: A Little Translation (20 points)

This question is designed to help you in memorizing the one-letter amino acid code as well as get acquainted with Python dictionaries. Dictionaries are very useful data structures that, like lists, can store large amounts of data. However, unlike lists, they may be indexed arbitrarily.

Consider the following example. You have a protein sequence, GGAYCFQWEN, and you have searched for that sequence in a protein sequence database. Suppose that, as of February 4, 2006, you received 10 hits in that database, and that you stored those results in a file called "foo.txt." One way to represent this information in Python would be to use a list. Then, you would establish that the first element in the list was the sequence, the second the number of hits, etc., like so:

```
seq = ['GGAYCFQWEN', 10, '02/04/2006', 'foo.txt']
```

From then on, you would have to remember the order of elements in your list.

Now suppose you were storing several hundred searches, and you had several hundred lines of code written over several days. Is the text file stored in position 2 or 3? Questions like these, for more complex data, may become burdensome, and it would be much more convenient to access this data using `seq['date']` as opposed to `seq[2]`.

Luckily, Provides an implementation of this solution, called a dictionary (also called a *hash table* in programming terms). Dictionaries in Python are defined using curly braces ({}). The following code fragment would be much more easily understood than the above example.

```
seq = {'sequence': 'GGAYCFQWEN',
       'hits'    : 10,
       'date'    : '02/04/2006',
       'file'    : 'foo.txt'}

print seq['sequence']
seq['file'] = 'bar.txt'
```

Dictionaries can also be easily modified. For example, suppose you wanted to add more data to your sequence record, describing the name of the database you were searching. In the first example, you would have to append a value on to your data list, but here, all you need to do is:

```
seq['database'] = 'SwissPROT'
```

Python will create the new key for you and assign its value to 'SwissPROT.' As you may be able to see at this point, a list of dictionaries, or even a dictionary of dictionaries, could be a very powerful and flexible data structure. We will work more with dictionaries later in the course, and you can read more about them in section 5.5 of the tutorial.

As the name implies, dictionaries may be used for simple translation. In this part of the assignment, we will implement a program that will translate a simple string of three-letter amino acid codes into a string of one-letter codes.

Since your final submission for this assignment will take as in put a string "Gly Gly Ala Tyr …" and output a string of the form "GGAY…," to complete your program you will need to first understand how to split a string into words (tokens).

a. [written] Download the string split code from the assignment website. Study the code carefully and try it on a few examples. Consider the line of code marked for question 3a (line 29). Explain how the code would behave differently if this line simply were:

   ```
   if char == ' ':
   ```

   Specifically, what inputs would be affected? (5 points)

b. [written] Consider the section of code on lines 35-36. Why is this code needed for correct function? What happens when it is removed? (5 points)

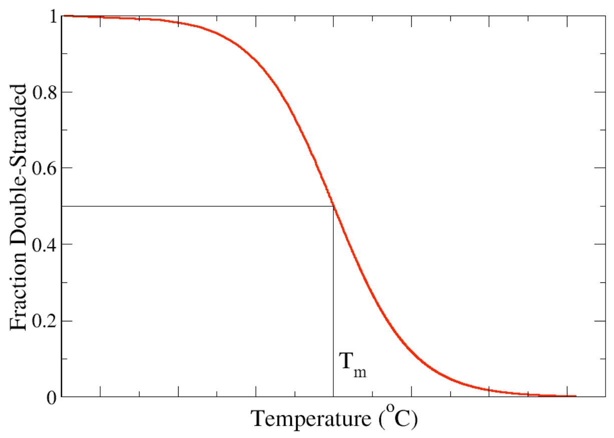
Note at the end of the string split code I provide two examples of how to split strings using built-in Python functions. You may use either of these methods to split the strings of amino acid names in part c of this question. **Bonus problem (5 points):** Convert the string-split example code into a function and use it in your submission for part c.

c. [programming] Download the amino acid translation code from the assignment website. Fill in the blanks for the translation dictionary, and write the code necessary to translate an arbitrary string of three-letter amino acid codes into one string of one-letter codes. (10 points)

Question 4: DNA Melting Temperature (Your first program from scratch) (20 points)

If you've completed the rest of the assignment, you should be ready for your first real program. There won't be any handholding for this one: you'll have to start from a blank screen and work from there. Your finished product should be well commented and easy to read, but you are free to solve it however you like.

As double-stranded DNA is heated, it undergoes a cooperative transition called melting where the two strands dissociate. This melting process can be described by a simple statistical mechanical model called helix-coil theory. If you haven't already, you will learn about helix-coil theory in your other biophysics classes. Helix-coil theory predicts that the shape of the DNA folding curve should be sigmoidal:



This is generally what's observed for most sequences. The melting temperature is defined as the temperature at which half of the DNA is double stranded and half is single stranded. While there are many equations for calculating the melting temperature for DNA sequences, we will use the simple Wallace rule that is often used for calculating the T_m for PCR primers. In this formula, the DNA melting temperature is incremented two degrees for every A or T base and four degrees for every G or C base.

Your assignment is to implement the Wallace formula. Your program should accept as input a DNA sequence (string) and print as output the predicted melting temperature. (20 points)