Homework 1 – 40 points total

This homework is due at the end of module 2 (the current module). The purpose is to make sure everyone can run Perl scripts, create simple scripts, and work with Perl constructs before Homework 2 is assigned.

Please turn in a zip of the following files:

- Hw_1_a.txt
- Your 4 scripts for parts B, C, D, and E.
- Take a screenshot for your output in E and include that file also.

Name your zip file <lastname_firstname>_HW1.zip

These are short scripts so I don't expect you to use the object-oriented style, but you can if you wish.

Part A (2 pts)

1. Download the zip of scripts and get perl_overview_basics.pl (from the video).
2. Run the script and redirect the output to a file. This can be done in a command window like so:
   `perl perl_overview_basics.pl > hw_1_a.txt`
3. You will turn in hw_1_a.txt as part of this homework.

Part B (2 pts)

Write a script that does the following:

1. Define a string variable and set it to "I have a string of words"
2. Use **split** to split the string on a single space. Split returns an array.
3. Use **join** to rejoin the new array using , [a comma] into a new string scalar variable. The string will have commas separating the items from the array.
4. Print your new string variable.

(Replacing a space with a , [comma] can be done several ways, please use split and join.)

Part C (6 points)

Write a script that does the following:

1. Reads in something you type at the keyboard (see <STDIN>). (Don't worry about removing newlines or carriage return line feeds in the input.)
2. Checks to see if the string begins with "Hi" using a **regular expression**.
3. If the string **begins** with Hi, change **ALL** occurrences of Hi in the input string to Hello and print the modified string. Use **regular expression substitution**. Look up the g modifier.
4. If the string does not being with Hi, call a **subroutine** that will **add** the characters "???" to the BEGINNING of the string and return the modified string. Then the script will print out the result of the subroutine call.

Be sure to try entering strings like

Hi there

Not Hi there [this string should not be modified because it doesn't begin with Hi]

Hi there. Hi again.

Part D (10 points)

Write a script that does the following:

1. Reads in the file HW_1_D.txt (included in the module).
2. Uses a **regular expression** to remove whitespaces before and after the text of each line.
3. Uses a **regular expression** to check that each line contains a well formed (or not) identifier according to this rule:
   2 letters followed by 4 or 5 numbers. See the use of **{}** to define how many characters.
4. If the identifier is well formed, write the identifier without any extra spaces (that is, the identifier with any whitespace around it removed) to a new file.
   a. Use **open** to open the file and **close** to close it. That is, do not use redirection to create the file.
5. If the identifier is not well formed, print an error message along with the malformed identifier.

NOTE: In HW_1_D.txt there are 3 good identifiers and 4 malformed ones. Don't worry about any blank lines in the input (or in your output file).

NOTE: HW_1_D.txt was created on a PC. Files created on PCs have 2 control characters at the end of each line – a carriage return and a line feed. This situation can cause an issue on UNIX-flavored systems because files on this OS use one character (a \n) only. When removing whitespaces, any carriage return or linefeed should also be eliminated in addition to any blank spaces.

Part E (20 points)

Write a script that does the following:

1. Create a subroutine (give it any name you like) that will take 3 inputs. Use a separate local variable declaration using "my" for each of the inputs (see the "shift" Perl reserved word or use @_ into separate variables).
2. In the subroutine, create a hash with the input so that it has the following key-value pairs and return the hash to the caller.

Key	**Value Type**
gene	String (the first input)
num_base_pairs	Integer (the second input)
organism	String (the third input)

3. In the main routine, call the subroutine 3 times using the following values. You do not have to do these calls in a loop, you can make 3 separate calls.
    - "ABC", 350, "An Org"
    - "DEF", 500, "DEF Org"
    - "GHI", 1000, "Last Org"

4. AND put the 3 returned hashes into an array (you can use "push"). Push using the hash reference (put \ in front of the variable name), like \%my_hash. Perl references were discussed in the Module 02 Lecture Content PDF and you can find information about them online.
5. You should end up with an array with each entry in the array being a reference to one of the little hash tables created by the subroutine. (3 hash elements in the array.) **Be careful**! Depending upon how you do this, make sure you don't overwrite your array elements with each new hash. Since you are working with references, you'll need to make sure you're referring to a new entity each time.
6. Loop through your array, using any technique you like. As you loop, print out a line of "----" then use the "**each**" Perl function to print out the keys and values of each little hash table. Separate the key and value with "=>". Remember to dereference the reference that you put into the array so you can recover the hash.
7. Your output should look like the following, but the ORDER of the keys for each little hash doesn't matter (because a hash is unordered). If you want to sort the keys, you can. But the data passed in with ABC will be the first set, DEF the second, and GHI the third. Note: I will check that your code follows the above instructions in addition to looking at your output.

```
----
gene => ABC
organism => An Org
num_base_pairs => 350
----
gene => DEF
organism => DEF Org
num_base_pairs => 500
----
organism => Last Org
num_base_pairs => 1000
gene => GHI
```

8. Take a screenshot of your terminal window showing your output.

NOTE: Perl has some picky punctuation. Be sure you keep track of when to use $, @, %, and [], or (), or {}. Look very closely at examples and follow them exactly until you understand how they work.

NOTE: You may use any reference book or website (but not a person) to obtain information to enable you to do your homework but please **credit in comments** all your sources!