CSC 526 Assignment 1

Q1 (Written):
Refer to the Lesk textbook, summarise the central dogma (of biology), in one page.
Estimated time: 30 minutes

Q2 (Programming **in PERL**):
Write a short program that **prompts the user** for a Fahrenheit temperature and prints the temperature in Celsius. Converting Fahrenheit temperature to Celsius can be accomplished by taking the Fahrenheit temperature and subtracting 32, and then multiplying by 5/9. For example, 95 degrees Fahrenheit is 35 degrees Celsius.
Estimated time: 10 minutes

Q3 (Programming **in PERL**):
Write a short program that uses **a loop** to display a table of the Fahrenheit temperatures -20 through 100 degrees and their equivalents in Celsius.
Estimated time: 10 minutes

Q4 (Programming **in PERL**):
Write a short program to store the total incomes of a bioinformatics research lab for each of 12 months into an **array** of floating point numbers. The program should calculate and return the following:

- the total income for the year
- the average monthly income
- the month with the most income
- the month with the least income

Estimated time: 30 minutes

Q5 (Programming **in PERL**):
Write a short program that uses the following **arrays**:

- employeeId. An array of 6 integers to hold a research lab's employee ID: (3561, 6782, 1280, 3345, 1235, 8777)
- hour. An array of 6 integers to hold the number of hours worked by each employee
- payRate. An array of 6 floating point numbers to hold the number of hours worked by each employee
- wage. An array of 6 floating point numbers to hold each employee's gross wages

The program is to relate data in each array through the index of the arrays. For example, number in element 0 of the hour array should be the number of hours worked by the employee whose ID is stored in element 0 of the employeeId array.
The program should:

- display each employee number and ask the user to enter that employee's hours and pay rate, store input in the arrays
- calculate the gross wages for the employee and store in the wages array
- display each employee's identification number, hours, payRate, and gross wages by reading the arrays
- do **input validation**: Do not accept negative values for hours or numbers less than 8.00 for pay rate (show/print test data to reflect this implementation)

Estimated time: 1 hour

Submission instructions:
Please submit a paper copy and an electronic copy.

Paper copy:

- Please submit the paper copy at the beginning of the class.
- Provide (create) **test input data** to all programming questions, and capture the related outputs as screen captures (or output files).
- Print the program source code files, test input data and **the output screen captures (or output files)**. If no output screen capture (or output file) is submitted, it would be assumed that the related program does not compile. If the print-out is not readable, **no mark will be awarded**.
- Identify each assignment question by writing the question number at the top of each page.
- Add the following statement to the first page of your submission: "I have abided by the UNCG Academic Integrity Policy on this assignment". Please write your full name and sign next to the statement. If the statement or the signature is not found, **75% of the possible points will be deducted**.

Electronic copy:

- Please submit a **lastname_firstname_assignment01.zip** (or lastname_firstname_assignment01.rar) file through the Blackboard Digital Dropbox. This zip (or rar) file should contain all submission files.
- Put the answers of all written questions in a **lastname_firstname_assignment01.doc** file.
- For programming questions, you only need to submit the *.pl files.
- Name each *.pl file according to the question number (e.g. Q1_*.pl).

Grading guidelines (programming questions):
Your programs will be judged on several criteria, which are shown below.

- Correctness (50%): Does the program compile correctly? Does the program do what it's supposed to do?
- Design (20%): Are operations broken down into functions / procedures in a reasonable way?
- Style (10%): Is the program indented properly? Do variables have meaningful names?
- Robustness (10%): Does the program handle erroneous or unexpected input gracefully?
- Documentation (10%): Do all program files begin with a comment that identifies the author, the contents, and the compiler used for that particular file? Are all the functions, procedures and data fields clearly documented? Are unclear parts of code documented?

A program that does not compile will get at most **50% of the possible points**.