Miami Dade College

GENERAL INFORMATION

Course Prefix/Number: **COP2004**	Course Title: **PERL PROGRAMMING I**
Number of Credits: 4 credits (2 hr. lecture; 2 hr. lab)	
Degree Type	☐ B.A. ☐ B.S. ☐ B.A.S ☒ A.A. ☒ A.S. ☐ A.A.S. ☐ C.C.C. ☒ A.T.C. ☐ V.C.C
Date Submitted/Revised:	Effective Year/Term: 2007-1
☒ New Course Competency ☐ Revised Course Competency	
Course Description (limit to 50 words or less): This course provides a practical introduction to PERL programming for the biology/bioinformatics student. Through lectures, real-world examples and extensive hands-on assignments, the student will acquire an understanding of the PERL syntax and use it to create and execute PERL modules that solve common bioinformatic programming demands. Lab fee. 3 hr. lecture; 2 hr. lab. Prerequisites: CGSXXXX Introduction to Bioinformatics, CIS2321 (Systems Analysis & Design).	
Prerequisite(s): CGSXXXX (Introduction to Bioinformatics), CIS2321	Corequisite(s):

Competency 1:
The student will demonstrate a generalized understanding of basic software engineering principles by:

1. Writing a requirements document detailing a bioinformatics need or problem that has to be addressed via programming.
2. Analyzing the requirements document.
3. Designing the solution using a modeling tool.
4. Practicing one of the development process methodologies.

Competency 2:
The student will demonstrate knowledge of the PERL environment and language by:

1. Installing PERL.
2. Obtaining, installing and using PERL library modules.
3. Using the on-line PERL documentation guides and reference materials.
4. Using a common text editor by which to create and edit PERL programs.
5. Utilizing the module development life-cycle by:
    a. Edit
    b. Save
    c. Run
    d. Validate
    e. Catalogue
6. Using and interpreting the output of the PERL debugger.
7. Using the command-line switches.
8. Manipulating the PERL programs' operating system flags.

Competency 3:
The student will demonstrate an understanding of the PERL syntax by:

1. Writing PERL modules using the standard declaratory syntax.

Revision Date 8/23/2007--4:23:43 PM

Approved By Academic Dean Date: ____________ Reviewed By Director of Academic Programs Date: ____________

2. Writing PERL modules using the object model syntax.
3. Using the STRICT construct.
4. Using the USE invocation of standard packages.

Competency 4:
The student will exhibit a practical understanding of PERL variables by:

1. Writing PERL modules that effectively use Scalars.
2. Writing PERL modules that effectively manipulate Arrays.
3. Writing PERL modules that effectively manipulate Hashes.
4. Writing PERL modules that de-reference Scalars, Arrays and Hashes.

Competency 5:
The student will exhibit a practical understanding of PERL iterative and control constructs by:

1. Writing PERL modules that employ conditional statements.
2. Writing PERL modules that use the looping structures:
   a. For
   b. While
   c. Foreach
3. Writing PERL modules that encompass subroutines.
4. Writing PERL modules that invoke subroutines and packages.
5. Developing PERL packages.

Competency 6:
The student will exhibit a practical understanding of PERL I/O interfaces by:

1. Writing PERL modules to accept input via the console.
2. Writing PERL modules to display output on the standard console.
3. Writing PERL modules to
   a. Read text files
   b. Write text files
   c. Update text files
4. Including the appropriate operating system syntax by which to access the files in specific directories with the appropriate permissions.
5. Creating the constructs to handle file exceptions.
6. Creating the constructs to notify file exception events.

Competency 7:
The student will demonstrate the ability to devise, develop, and deploy PERL regular expressions by:

1. Incorporating the *chop*, *chomp*, *length* and *substr* constructs in a module/program.
2. Incorporating the PERL matching operators in a construct.
3. Employing modifiers in an expression.
4. Employing delimiters, special characters and meta-characters in an expression.
5. Using *greedy* versus *not greedy* syntax.
6. Combining constructs, modifiers, and delimiters to write regular expressions that search strings of data for specific values.
7. Combining constructs, modifiers, and delimiters to write regular expressions that search strings of data for patterns.
8. Combining constructs, modifiers, and delimiters to write regular expressions that replace characters within strings of data.

9. Combining constructs, modifiers, and delimiters to write regular expressions that replace patterns within strings of data.
10. Combining constructs, modifiers, and delimiters to write regular expressions that search variant sets of strings of data for patterns.

Competency 8:
The student will develop a PERL package encapsulating bioinformatics-specific functions by:

1. Writing a module to read, store, and manipulate text files containing DNA sequences.
2. Writing a module to perform sequence analysis.
3. Writing a module to transcribe programmatically DNA to RNA.
4. Writing a module to create the DNA matching strand (base pairs) from one strand.
5. Writing a module to translate condons to the amino acids.
6. Packaging these modules.

Competency 9:
The student will develop a bioinformatics-specific PERL program to search for motifs in DNA or protein by:

1. Reading a protein sequence from a text file.
2. Loading the entirety of the sequence into a string.
3. Accepting the search motif via user input.
4. Writing a regular expression by which to search for the motif (user input of variant length) throughout the protein sequence.
5. Counting each type of nucleotide based on its base.

Competency 10:
The student will develop a bioinformatics-specific PERL program to simulate DNA mutations and randomness by:

1. Creating a random number sequence through writing the code by which to seed and display the results of the *srand* function.
2. Loading a DNA sequence into an array or hash.
3. Using the *srand* function, randomly select a position or positions in the strand.
4. Model a mutation by using randomly generated numbers to select a nucleotide in the DNA strand, the use of random numbers to mutate into a (random) neucleotide.
5. Using the *srand* function, generate random numbers to generate DNA sequence strings.
6. Applying an iterative construct to DNA mutation programming so as to simulate mutations over time.