Programming and Software Development (PSD)

Course Descriptions

Fundamentals of Information Systems Technology
This course is a survey of computer technologies. This course may include computer history, ethics of computer and network use, web design, introduction to graphics, animation, computer hardware and operating systems, elementary networking, troubleshooting, and programming. (The student should have prior knowledge of an office suite to include but not limited to word processing, spreadsheet, database and presentation software.)

PSD1
This course provides the beginning programmer with a solid foundation in programming fundamentals, using visualization and application. Designed for beginners with little or no previous programming experience, this course teaches students the fundamentals behind all programming languages. By putting standard concepts, like input, output, selection, and repetition, at the forefront, instead of focusing solely on a specific language, students will gain knowledge and insight that is easily transferable to other languages. (Outcomes: 1 – 7, 11 - 14)

PSD2
This course provides the beginning programmer with object-oriented programming concepts. The course reviews basic programming concepts such as structure, decision making, looping, arrays, and method calling and enforces good style and logical thinking. Advanced topics include inheritance, exceptions, GUIs, events, and files. (Outcomes: 7 – 10, 11 - 14)

PSD3
This course is designed to be a capstone experience for the student where they will apply knowledge and skills learned throughout their course of study as they apply advanced programming and development topics. Students in this course will also investigate current and emerging technologies. This course will also provide an introduction to Systems Analysis and Design. Topics include analyzing cases, requirements modeling, data and process modeling, and development strategies. Students also learn about output and user interface design, data design, systems architecture and implementation, and systems operation, support and security. Students will take the Technical Skills Assessment in this course. (Outcomes: 1 - 14)

Recommended courses include: Web Design & Development 1 & 2

Outcomes and Learning Objectives

Outcome 1: Students will develop critical thinking and problem solving skills as they apply to programming.
- Understand basic programming principles
- Understand procedural and object-oriented programming
- Understand the features of object-oriented programming languages
- Write a program that produces output
- Understand how to select identifiers to use within your programs
- Improve programs by adding comments
- Write and run a program

Outcome 2: Students will demonstrate ability to use variables, data types, and string manipulation to solve computer problems programmatically.
- Demonstrate the process of declaring variables
- Display variable values
• Understand the use of integral data types
• Understand the use of floating-point data types
• Use arithmetic operators
• Understand the boolean data type
• Understand numeric type conversion
• Understand the char data type
• Understand the string data type
• Define named constants and enumerations

**Outcome 3:** Students will demonstrate understanding of and will effectively use selection structures to add logic to their programs.
  • Understand logic-planning tools and decision making
  • Make decisions using the if statement
  • Make decisions using the if-else statement
  • Use compound expressions in if statements
  • Make decisions using the switch statement
  • Use the conditional operator
  • Use the NOT operator
  • Understand how to avoid common errors when making decisions

**Outcome 4:** Students will demonstrate ability to test, debug and validate programming applications.
  • Locate a logic error by stepping through the code
  • Locate logic errors using breakpoints
  • Fix syntax and logic errors
  • Select appropriate test data for an application

**Outcome 5:** Students will differentiate between the various types of repetition structures and use each repetition structure appropriately in their programs.
  • Use the loop structure
  • Create loops using the while statement
  • Create loops using the for statement
  • Create loops using the do statement
  • Use nested loops
  • Use accumulators
  • Understand how to improve loop performance

**Outcome 6:** Students will use methods to increase functionality and to modularize their programs.
  • Learn about methods and implementation hiding
  • Write methods with no parameters and no return value
  • Write methods that require a single argument
  • Write methods that require multiple arguments
  • Write a method that returns a value
  • Pass an array to a method
  • Overload methods
  • Demonstrate how to avoid ambiguous methods
  • Use optional parameters

**Outcome 7:** Students will demonstrate understanding of arrays and structures and will apply concepts in their programs.
  • Declare an array and assign values to array elements
  • Access array elements
  • Search an array using a loop
  • Use multidimensional arrays

**Outcome 8:** Students will demonstrate understanding of Object-Oriented Programming Concepts.
  • Understand class concepts
• Create classes from which objects can be instantiated
• Create objects
• Create properties, including auto-implemented properties
• Use public fields and private methods
• Learn about the this reference
• Write constructors and use them
• Use object initializers
• Overload operators
• Declare an array of objects and use the sorting methods with them
• Write destructors
• Understand inheritance
• Extend classes
• Override base class methods
• Understand how a derived class object “is an” instance of the base class
• Understand the Object class
• Work with base class constructors
• Create and use abstract classes
• Create and use interfaces
• Use extension methods
• Recognize inheritance in GUI applications and understand the benefits of inheritance

Outcome 9: Students will understand the importance of using exception handling in their programs.
• Understand traditional and object-oriented error-handling methods
• Cast data types
• Catch multiple Exceptions
• Use the finally block
• Handle Exceptions thrown from outside methods
• Trace Exceptions through the call stack
• Create your own Exception classes
• Re-throw Exceptions

Outcome 10: Students will discover and use event handlers in their programs.
• Understand event handling
• Understand delegates
• Declare your own events and handlers and use the built-in event handlers
• Handle Control component events
• Handle mouse and keyboard events
• Manage multiple Controls
• Understand how to continue your exploration of Controls and Events

Outcome 11: Students will understand systems planning, development and apply concepts and principles.
• Describe the information systems development life cycle (SDLC).
• Discuss how to evaluate off-the-shelf software.
• Explain reuse and its role in software development.
• Describe the skills required to be an effective project manager.
• List and describe the skills and activities of a project manager during project initiation, project planning, project execution, and project closedown.
• Describe the steps involved when identifying and selecting projects and initiating and planning projects.
• Explain the need for and the contents of a Project Scope Statement.
• List and describe various methods for assessing project feasibility.

Outcome 12: Students will demonstrate competency with systems analysis tools and concepts.
• Describe options for designing and conducting interviews and develop a plan for conducting an interview to determine system requirements.
• Explain the advantages and pitfalls of observing workers and analyzing business documents to determine system requirements.
• Participate in and help plan a Joint Application Design session.
• Use prototyping during requirements determination.
• Select the appropriate methods to elicit system requirements.
• Understand how requirements determination techniques apply to development of Internet applications.
• Understand the logical modeling of processes through studying examples of data-flow diagrams, pseudo code, and flowcharts.

Outcome 13: Students will demonstrate knowledge of application design principles.
• Explain the process of designing interfaces and dialogues and the deliverables for their creation.
• Describe and apply the general guidelines for interface design, including guidelines for layout design, structuring data-entry fields, providing feedback, and system help.
• Concisely define each of the following key database design terms: relation, primary key, functional dependency, foreign key, referential integrity, field, data type, null value, denormalization, file organization, index, and secondary key.
• Explain the role of designing databases in the analysis and design of an information system.
• Transform an entity-relationship (E-R) diagram into an equivalent set of well-structured (normalized) relations.
• Merge normalized relations from separate user views into a consolidated set of well-structured relations.
• Choose storage formats for fields in database tables.
• Translate well-structured relations into efficient database tables.
• Explain when to use different types of file organizations to store computer files.
• Describe the purpose of indexes and the important considerations in selecting attributes to be indexed.

Outcome 14: Students will demonstrate knowledge of application implementation and identify the need for ongoing application support.
• Describe the process of coding, testing, and converting an organizational information system and outline the deliverables and outcomes of the process.
• List the deliverables for documenting the system and for training and supporting users.
• Compare the many modes available for organizational information system training, including self-training and electronic performance support systems.
• Discuss the issues of providing support for end users.
• Explain why application implementation sometimes fails.
• Describe several factors that influence the cost of maintaining an application.
Program Criteria

Program Cluster: Information Systems Technology
Program: Programming and Software Development

Program Requirements –
In order that a student might be considered a concentrator and be considered for Tech Prep articulation to a qualifying postsecondary program the student should:
Complete the secondary Programming and Software Development Program** consisting of:

100012 TI 0550 Fundamentals of Information Systems Technology
101522 TI 0557 Programming & Software Development I
101524 TI 0558 Programming & Software Development II
101523 TI 0610 Programming & Software Development III (CAPSTONE)

**See corresponding program Outcomes and Objectives document for program outcomes details.

Upon completion of the Capstone course the student would be required to take a Technical Skill Assessment, To Be Determined (4/9/2013). This will complete the concentrator requirements under Carl Perkins for a program of study. This assessment will test the student’s competency in the following areas:

- CTECS to develop an assessment that will allow a student to choose the programming language used in the questions.

Upon completion and passing of the Technical Skills Assessment the student would qualify for Tech Prep articulation at the below listed colleges for the courses described.

<table>
<thead>
<tr>
<th>College</th>
<th>Course Number/Name</th>
<th>Credits</th>
</tr>
</thead>
<tbody>
<tr>
<td>LCSC</td>
<td>CITPT 106 Intro to Comp Programming</td>
<td>3</td>
</tr>
<tr>
<td>CSI</td>
<td>CISW 125 Intro to Programming</td>
<td>3</td>
</tr>
<tr>
<td>CWI</td>
<td>WEBD 130 Intro to Programming</td>
<td>4</td>
</tr>
<tr>
<td>EITC</td>
<td>ELC 203 Intro to Comp Programming</td>
<td>3</td>
</tr>
</tbody>
</table>