Multiple Category Scope and Sequence: Scope and Sequence Report For Course Standards and Objectives, Content, Skills, Vocabulary

Wednesday, August 20, 2014, 12:22PM

	Unit	Course Standards and Objectives	Content	Skills	Vocabulary
District Intermediate **Computer Programming I (11.0201) (District)** 2014-2015 **Collaboration**	**History of Computing** (Week 1, 1 Week)	UT: CTE: Information Technology, UT: Grades 9-12, Computer Programming I Standard 6 Students will develop an awareness of career opportunities in the Computer Programming/Software Engineering industry and of its history. ▪ Objective 5 Discuss relevant history of software development a. Discuss relevant history of computer technology b. Identify key points in the history of the Computer Programming/Software Engineering industry	Students will know the pre-historic/mother languages. Students will be able to list major technology advancements. Students will be able to identify historic figures in technology and describe their contribution.	Students will be able to speak and write about major technology advancements.	IP address. Protocols. Hardware components (CPU, CRT, Monitor) Language type names (OOPs). Windows Forms application form web form class namespace C# compiler Microsoft Intermediate Language (MSIL) assembly solution file Form Designer Toolbox

				Properties window Code Editor web browser
C# IDE Hello World (Week 2, 1 Week) Standard 1 - Objective 1 - **Demonstrate knowledge of external and internal computer hardware.** How to use the control panel for hardware settings. How to change hardware settings. How to install and remove software applications using the control panel How to check for memory and CPU sizes. How to check for disk space usage and unused space available. How to convert decimal values to binary values. Objective 2 - **Demonstrate knowledge of software concepts.** How is software different from hardware. How to run and install application software. How to use web based software. How to use Utility software. How a C# application is compiled and run. How is a low level language different from a high level language. Objective 3 - **Develop the ability to use a current operating system.**	UT: CTE: Information Technology, UT: Grades 9-12, Computer Programming I Standard 1 Students will be familiar with and use a programming environment. ▪ Objective 1 Demonstrate knowledge of external and internal computer hardware. a. Describe the functions of basic computer hardware devices (monitor, printer, CD-ROM drive, floppy drive, keyboard, mouse, adapters, other devices). b. Describe the functions of the internal components of computers (CPU, RAM, ROM, Motherboard, graphics card, hard drive). c. Utilize the binary numbering system (translate from binary to decimal and vice-versa). ▪ Objective 2 Demonstrate knowledge of software concepts. a. Define the distinction between computer software and hardware. b. Identify software categories such as Application Software, Web Based Software, OS, Utility Software (anti virus, system tools). c. Describe the difference between an interpreted language vs a compiled language d. Describe the difference between a low level and high level language ▪ Objective 3 Develop the ability to use a current operating system. a. Demonstrate how to open	Students will be able to use the control panel for hardware settings. Students will be able to change hardware settings. Students will be able to install and remove software applications using the control panel Students will be able to check for memory and CPU sizes. Students will be able to check for disk space usage and unused space available. Students will be able to convert decimal values to binary values. Objective 2**.** Students will be able to software different from hardware. Students will be able to run and install application software. Students will be able to use web based software. Students will be able to Utility software. Students will be able to run and debug a C# applications. is compiled and run. Students will be able to differentiate a low level language different from a high level language. Students will be able to start Visual Studio. Students will be able to open or close an	block of code syntax error build error comment collapse expand code snippet build a project run a project runtime error bug debug binary numbers

and save files.
b. Demonstrate how to move, rename, copy, compress and delete files.
c. Demonstrate how to display and print files.
d. Create and use appropriate directory and path structures
e. Demonstrate how to execute a program.

- Objective 4
Demonstrate the ability to use the editor to write code.
a. Demonstrate the process of selecting a block of text.
b. Demonstrate how to move, copy, and delete blocks of text.
- Objective 5
Demonstrate the ability to compile, debug and execute programs.
a. Demonstrate how to use the editor to compile and run programs.
b. Understand the difference between syntax, run-time, and login errors.
c. Demonstrate how to debug programs.
d. Optional — Use a debugger to set break-points, and step through code to track down errors at runtime

How to start Visual Studio.

How to open or close an existing project.

How to use the Form Designer.

How to use the Code Editor.

How to use the Visual Studio Explorer.

Objective 4 - **Demonstrate the ability to use the editor to write code.**

How to code with readable style.

How to code comments.

How to work with the Text Editor toolbar.

How to collapse or expand blocks of code.

Objective 5 - **Demonstrate the ability to compile, debug and execute programs.**

How to run a project in Visual Studio.

How to test a project.

How to debug runtime errors.

Chapter 1 History of Computing

existing project.

Students will be able to use the Form Designer.

Students will be able to use the control panel for hardware settings.

Students will be able to change hardware settings.

Students will be able to install and remove software applications using the control panel

Students will be able to check for memory and CPU sizes.

Students will be able to check for disk space usage and unused space available.

Students will be able to convert decimal values to binary values.

Objective 3 - **Develop the ability to use a current operating system.**

Students will be able to start Visual Studio.

Students will be able to open or close an existing project.

Students will be able to use the Form Designer.

Students will be able to use the control panel for hardware settings.

Students will be able to change hardware settings.

Students will be able to install and remove software applications using the control panel

Students will be able to check for memory and CPU sizes.

Students will be able to check for disk space

usage and unused space available.

Students will be able to convert decimal values to binary values.

Objective 4 - **Demonstrate the ability to use the editor to write code.**

Students will be able to code with readable style.

Students will be able to code comments.

Students will be able to work with the Text Editor toolbar.

Students will be able to use the control panel for hardware settings.

Students will be able to change hardware settings.

Students will be able to install and remove software applications using the control panel

Students will be able to check for memory and CPU sizes.

Students will be able to check for disk space usage and unused space available.

Students will be able to convert decimal values to binary values.

Objective 3 - **Develop the ability to use a current operating system.**

How to start Visual Studio.

How to open or close an existing project.

How to use the Form Designer.

How to use the Code Editor.

How to use the Visual Studio Explorer.

Objective 4 - **Demonstrate the ability to use the editor to write code.**

How to code with readable style.

How to code comments.

How to work with the Text Editor toolbarHow to use the control panel for hardware settings.

How to change hardware settings.

How to install and remove software applications using the control panel

How to check for memory and CPU sizes.

How to check for disk space usage and unused space available.

How to convert decimal values to binary values.

Objective 2 - **Demonstrate knowledge of software concepts.**

How is software different from hardware.

How to run and install application software.

How to use web based software.

How to use Utility software.

How a C# application is compiled and run.

How is a low level language different from a high level language.

Objective 3 - **Develop the ability to use a current operating system.**

			How to start Visual Studio. How to open or close an existing project. How to use the Form Designer. How to use the Code Editor. How to use the Visual Studio Explorer. Objective 4 - **Demonstrate the ability to use the editor to write code.** How to code with readable style. How to code comments. How to work with the Text Editor toolbar	
Variables and I/O (Week 3, 3 Weeks)	UT: CTE: Information Technology, UT: Grades 9-12, Computer Programming I Standard 2 Students will employ accepted programming methodology. ▪ Objective 1 Demonstrate the ability to use good programming style. a. Demonstrate how to use white space properly. b. Employ proper naming conventions (such as Camel Case and Pascal Case). c. Construct programs with meaningful identifiers. ▪ Objective 2 Follow the major steps of a Software Development Life Cycle (SDLC). a. Prepare specifications and requirements for computer programs. b. Design solutions using algorithms such as flow charts, pseudo code, and basic UML. c. Implement the code for a program. d. Test programs for effectiveness and completeness. e. Provide documentation for a	SStandard 1 Objective 1 - Demonstrate the ability to use good programming style: How to code with readable style. How to code comments. How to Work with the text with the text editor toolbar. How to expand or collapse blocks of code. Standard 2 - Students will employ accepted programming methodology. Objective 1 - **Demonstrate the ability to use good programming style.** a. Demonstrate how to use white space properly. b. Employ proper naming conventions (such as Camel Case and Pascal Case). c. Construct programs with meaningful identifiers. Objective 2 - **Follow the major steps of a Software Development Life Cycle (SDLC).** How to create a user requirements document.	Students will be able to read and write comments with correct syntax. Students will be able to execute and debug programs. Students will be able to use proper camel like notation. Students will be able to Identify keywords, identifiers, operators, operands, and literals. Students will be able to Identify the entry-point of a program. Students will be able to Identify statements and expressions in a program. Students will be able Identify subroutines in a program.	build-in data types bit byte integer string ASCII character boolean value variable declare initialize camel notation literal

program (such as internal and external documentation).

- Objective 3
  Identify the syntactical components of a program
  a. Identify keywords, identifiers, operators, operands, and literals
  b. Identify the entry-point of a program
  c. Identify statements and expressions in a program
  d. Identify subroutines in a program

Standard 3
Students will properly use language-fundamental commands and operations.

- Objective 1
  Demonstrate the ability to use basic elements of a specific language.
  a. Write programs using a language-specific template.
  b. Declare, initialize, and assign values to constants and variables.
  c. Output text with formatting.
  d. Demonstrate the ability to use input/output commands.
  e. Output values stored in identifiers.
- Objective 2
  Employ basic arithmetic expressions in programs.
  a. Use basic arithmetic operators (addition, subtraction, modulus, multiplication, division)
  b. Understand order of operation of expressions
  c. Write expressions that mix floating-point and integer expressions.
  d. Write expressions to accumulate values.
- Objective 3
  Demonstrate the ability to use data types in programs.
  a. Declare and use primitive data types (integer, floating point, Boolean)

How to use pseudo code.

How to read and create a flow chart.

How to write a program and execute.

How to test and review output.

How to create a users guide.

Objective 3 - **Identify the syntactical components of a program.**
a. Identify keywords, identifiers, operators, operands, and literals
b. Identify the entry-point of a program
c. Identify statements and expressions in a program
d. Identify subroutines in a program

Standard 3 - Students will properly use language-fundamental commands and operations.

Objective 1 - **Demonstrate the ability to use basic elements of a specific language.**

How to work with built-in value types.

How to declare and initialize variables.

How to declare and initialize constants.

Objective 2 - **Employ basic arithmetic expressions in programs.**

How to code arithmetic expressions.

How to code assignment statements.

How to work with order of precedence.

Objective 3 - **Demonstrate the ability to use data types in programs.**

How to use type casting.

operand

binary operator

assignment statement

assignment operator

order of precedence

type casting

concatenate

null value

argument

	b. Declare and use reference (non-primitive) types c. Declare and use constants. d. Optional — Declare and use enumerators as a list of constants ▪ Objective 4 Demonstrate the ability to use strings in programs. a. Declare string identifier. b. Input string identifiers. c. Output string identifiers.	How to use the Math class. How to use methods to convert data types. How to use methods to convert numbers to formatted strings. Objective 4 - **Demonstrate the ability to use strings in programs.** How to declare and initialize strings. How to join and append strings. How to include special characters in strings.		
Decision (Week 6, 4 Weeks)	UT: CTE: Information Technology, UT: Grades 9-12, Computer Programming I Standard 4 Students will properly employ control structures. ▪ Objective 1 Demonstrate the ability to use relational and logical operators in programs. a. Compare values using relational operators. b. Form complex expressions using logical operators. ▪ Objective 2 Demonstrate the ability to use decisions in programs. a. Employ simple IF structures. b. Use IF-ELSE structures. c. Write programs with nested IF-ELSE structures. d. Make multiple-way selections (switch, case).	How to use C# relational operators. How to use C# logical operators. How to code "if else" statements. Debugging techniques for programs.	Students will be able to use C# relational operators. Students will be able to use C# logical operators. Students will be able to code "if else" statements. Students will be able to use debugging techniques for C# programs.	1. Control Structures 2. Relational Operators 3. Logical Operators 4. Short-Circuit Operators
Loops (Week 10, 2 Weeks)	UT: CTE: Information Technology, UT: Grades 9-12, Computer Programming I Standard 4 Students will properly employ control structures. ▪ Objective 3 Demonstrate the ability to use loops in programs. a. Use initial, terminal, and	Objective 3 - Demonstrate the ability to use loops in programs: How to code "while", "do while" and "for" loops. How to code for loops How to use loops with break and continue	Students will be able to code "while", "do while" and "for" loops. Students will be able to code for loops Students will be able to use loops with break and continue statements. Students will be able to use debugging	Control Structures Relational Operators Logical Operators

	incremental values in loops. b. Construct both pre-test and post-test loops. c. Demonstrate how to use counted loops. d. Describe the use of flagged (sentinel-controlled) loops. e. Utilize nested loops. f. Explain how to avoid infinite loops. g. Accumulate running totals using loops.	statements. How to use debugging techniques for programs with loops.	techniques for programs with loops.	Short-Circuit Operators Boolean Expressions if-else statement block scope switch statement case label default statement break statement iteration structure do-while statement continue statement breakpoint jump statement
Arrays and Strings (Week 12, 2 Weeks)	UT: CTE: Information Technology, UT: Grades 9-12, Computer Programming I Standard 7 Students will employ arrays. ▪ Objective 1 Demonstrate the ability to use arrays in programs. a. Declare arrays all applicable types. b. Initialize arrays. c. Input data into arrays. d. Output data from arrays. e. Perform operations on	Standard 7 - Students will employ arrays. Objective 1 - **Demonstrate the ability to use arrays in programs.** How to create an array How to assign values to the elements of an array. How work with an array.	Students will be able to create an array Students will be able to assign values to the elements of an array. Students will be able to work with an array. Students will be able to work with a list. Students will be able to with a sorted list. Students will be able to work with queues and	array element length size one-dimensional array

	arrays. f. Perform sequential searches on arrays. ▪ Objective 2 Demonstrate the ability to use dynamic arrays (i.e. vectors, arraylists, or generic lists) a. Declare a dynamic array b. Add and remove items from the array c. Output data from arrays. d. Perform operations on arrays. e. Iterate through the loop (i.e. foreach loop) ▪ Objective 3 Demonstrate the ability to use strings in programs. a. Compare string identifiers. b. Find the length of a string. c. Copy part or all of string identifiers into other strings. d. Concatenate string identifiers. e. Locate and delete sub-string positions. f. Insert strings into other strings.	How to work with a list. How to work with a sorted list. How to work with queues and stacks. How to work with an array list. Objective 2 - **Demonstrate the ability to use dynamic arrays (i.e. vectors, arraylists, or generic lists)** How to use the Array class. How to refer to and copy arrays. How to use foreach loops to work with arrays. How to code methods that work with arrays. Objective 3 - **Demonstrate the ability to use strings in programs.** How to work with strings. How to use properties and methods of the String class. How to use code examples that work with strings. How to use the StringBuilder class. How to use methods for validating user entries.	stacks. Students will be able to work with an array list. Objective 2 - **Demonstrate the ability to use dynamic arrays (i.e. vectors, arraylists, or generic lists)** v use the Array class. Students will be able to refer to and copy arrays. Students will be able to use foreach loops to work with arrays. Students will be able to code methods that work with arrays. Objective 3 - **Demonstrate the ability to use strings in programs.** Students will be able to work with strings. Students will be able to use properties and methods of the String class. Students will be able to use code examples that work with strings. Students will be able to use the StringBuilder class. Students will be able to use methods for validating user entries.	index upper bound foreach statement foreach loop list sorted list queue stack first-in first-out FIFO last-in last-out LIFO array list
Class and Objects (Week 14, 2 Weeks)	UT: CTE: Information Technology, UT: Grades 9-12, Computer Programming I Standard 8 Students will properly employ object-oriented programming techniques. ▪ Objective 1 Demonstrate the ability to use classes. a. Instantiate objects. b. Use object data members. c. Use object member	Standard 8 - Objective 1 - **Demonstrate the ability to use classes:** How classes can be used to structure an application. How to define members within a class. How to instantiate objects.	Students will be able to use classes that can be used to structure an application. Students will be able to define members within a class. Students will be able to instantiate objects. Students will be able to add a class file to a project.	Instantiate Constructor Code Stubs Property Method

	functions (methods). ▪ Objective 2 Demonstrate the ability to create user-defined classes. a. Create and use data members. b. Create a constructor to initialize the data members. c. Create and use instance functions (methods). ▪ Objective 3 Demonstrate proper design principles with classes a. Create classes that are well encapsulated (data members private). b. Properly use modifiers and accessors (getters and setters). c. Understand private and public modifiers	How to add a class file to a project. How to code fields. How to code properties. Objective 2 - **Demonstrate the ability to create user-defined classes:** How to code methods How to code constructors. How to code static members. How to create code stubs. Objective 3 - **Demonstrate proper design principles with classes:** How to operate a Product Maintenance application. How to code for validation. How to create a structure. How to use a structure. How to read/write property.	Students will be able to code fields. Students will be able to code properties. Objective 2 - **Demonstrate the ability to create user-defined classes:** Students will be able to code methods Students will be able to code constructors. Students will be able to code static members. Students will be able to create code stubs. Objective 3 - **Demonstrate proper design principles with classes:** Students will be able to operate a Product Maintenance application. Students will be able to code for validation. Students will be able to create a structure. Students will be able to use a structure. Students will be able to change the read/write property attributes.	Delegate Event Field Constant Indexer Operator Class encapsulation reference type get accessor set accessor

Final Project (Week 16, 2 Weeks)	UT: CTE: Information Technology, UT: Grades 9-12, Computer Programming I Standard 1
	Students will be familiar with and use a programming environment.
	▪ Objective 3
	Develop the ability to use a current operating system.
	a. Demonstrate how to open and save files.
	b. Demonstrate how to move, rename, copy, compress and delete files.
	c. Demonstrate how to display and print files.
	d. Create and use appropriate directory and path structures
		Objective 1 - **Identify personal interests and abilities related to Computer Programming/Software Engineering careers**
		How to identify personal creative talents.
		How to identify technical/programming talents.
		How to identify organizational and leadership skills.
			▪ 1 Provide Overview of 6 models of the Software Development Life Cycle.
			▪ 2 Discuss Software development activities
			  ▪ Planning
			  ▪ Implementation, testing and documenting
			  ▪ Deployment and maintenance
			▪ 3 Discuss Software development models by providing a definition of each model.
			  ▪ Waterfall model
			  ▪ Spiral model
			  ▪ Iterative and incremental development
				portfolio
				Student Education Occupation Plan (SEOP)
				software analyst
				software engineer
				multimedia applications

e. Demonstrate how to execute a program.

- Objective 5
  Demonstrate the ability to compile, debug and execute programs.
  a. Demonstrate how to use the editor to compile and run programs.
  b. Understand the difference between syntax, run-time, and login errors.
  c. Demonstrate how to debug programs.
  d. Optional — Use a debugger to set break-points, and step through code to track down errors at runtime

Standard 6
Students will develop an awareness of career opportunities in the Computer Programming/Software Engineering industry and of its history.

- Objective 1
  Identify personal interests and abilities related to Computer Programming/Software Engineering careers
  a. Identify personal creative talents
  b. Identify technical/programming talents
  c. Identify organizational and leadership skills
  d. Explore aptitude for innovation
  e. Determine aptitude for working as a member of a Computer Programming/Software Engineering team
- Objective 2
  Identify Computer Science career fields
  a. Understand the work of a Software Engineer
  b. Understand what a Systems Analyst does
  c. Understand the kind of work performed by a Applications Programmer (Gaming,

How to explore aptitude for innovation.

How to determine aptitude for working as a member of a Computer Programming/Software Engineering team.

Objective 2 - **Identify Computer Science career fields**

How to work as a Software Engineer.

How to work as Systems Analyst.

How to perform work like an Applications Programmer (Gaming, Multimedia Etc.)

Objective 3 - **Investigate career opportunities, trends, and requirements related to Computer Programming/Software Engineering careers**

How to identify the members of a Computer Programming/Software Engineering team: Team Leader, Analyst, Sr. Developer, Jr. Developer, and Client/Subject, Matter Expert.

How to work with a Computer Programming/Software Engineering team.

How to analyze trends associated with Computer Programming/Software Engineering careers.

How to develop a realistic Student Education Occupation Plan (SEOP) to help guide further educational pursuits.

Objective 4 - **Identify factors for employ ability and advancement in Computer Programming/Software Engineering careers**

How to survey existing Computer Programming/Software Engineering businesses

- Agile development
- Rapid application development
- Code and fix

- 4 Provide Process improvement models by drawing them on the board - graphic.

project management

agile

scope creep

backlog

sprint

statement of work

Multimedia Etc.)

- Objective 3
  Investigate career opportunities, trends, and requirements related to Computer Programming/Software Engineering careers
  a. Identify the members of a Computer Programming/Software Engineering team: Team Leader, Analyst, Sr. Developer, Jr. Developer, and Client/Subject Matter Expert
  b. Describe work performed by each member of the Computer Programming/Software Engineering team
  c. Investigate trends associated with Computer Programming/Software Engineering careers
  d. Develop a realistic Student Education Occupation Plan (SEOP) to help guide further educational pursuits
- Objective 4
  Identify factors for employ ability and advancement in Computer Programming/Software Engineering careers
  a. Survey existing Computer Programming/Software Engineering businesses to determine what training is required
  b. Survey universities and colleges to determine higher education options
  c. Develop employability competencies/characteristics: responsibility, dependability, respect, and cooperation
  d. Achieve high standards of personal performance
  e. Develop a positive work ethic
  f. Compile a portfolio of the individual and group programs developed during the course
- Objective 5
  Discuss relevant history of

to determine what training is required.

How to survey universities and colleges to determine higher education options.

How to develop employability competencies/characteristics: responsibility, dependability, respect, and cooperation skills.

How to achieve high standards of personal performance.

How to develop a positive work ethic.

How to compile a portfolio of the individual and group programs developed during the course.

Objective 5 - **Discuss relevant history of software development**

How to discuss relevant history of computer technology.

How to identify key points in the history of the Computer Programming/Software Engineering industry.

- 1 Provide Overview
- 2 Review Software development activities
  - Planning
  - Implementation, testing and documenting
  - Deployment and maintenance
- 3 Discuss Software development models
  - Waterfall model
  - Spiral model
  - Iterative and incremental development
  - Agile development
  - Rapid application

	software development a. Discuss relevant history of computer technology b. Identify key points in the history of the Computer Programming/Software Engineering industry	development ▪ Code and fix ▪ 4 Process improvement models ▪ 5 Formal methods ▪ 6 Review Development methods and Related Subjects ▪ 7 Provide References ▪ 8 Use and compile external links		
Ethical and Social Computing (Week 18, 1 Week)	UT: CTE: Information Technology, UT: Grades 9-12, Computer Programming I Standard 5 Students will demonstrate knowledge of current ethical issues dealing with computers and information in society. ▪ Objective 1 Understand ethical responsibility of software developers a. Explain the ethical reasons for creating reliable and robust software. b. Explain the impact software can have on society. c. Show how security concerns can be addressed in a program. ▪ Objective 2 Demonstrate knowledge of the social and ethical consequences of computers. a. Describe how computer-controlled automation affects a workplace and society. b. Explain the ramifications of society's dependence on computers. c. Identify advantages and disadvantages of changing workplace environments. ▪ Objective 3 Demonstrate knowledge of the right to privacy. a. Explain how computers can compromise privacy. b. Exhibit knowledge of privacy laws. c. Describe responsibilities of people who control computer information.	Objective 1 - **Understand ethical responsibility of software developers** The ethical reasons for creating reliable and robust software. Impact software can have on society. Security concerns can be addressed in a program. Objective 2 - **Demonstrate knowledge of the social and ethical consequences of computers.** Computer-controlled automation affects a workplace and society. Ramifications of society's dependence on computers. Advantages and disadvantages of changing workplace environments. Objective 3 - **Demonstrate knowledge of the right to privacy.** Computers can compromise privacy. Knowledge of privacy laws. Responsibilities of people who control computer information.	Students will be able to discuss and write about the ethical reasons for creating reliable and robust software. Impact software can have on society. Students will be able to discuss and write about the security concerns that can be addressed in a program. Objective 2 - **Demonstrate knowledge of the social and ethical consequences of computers.** Students will be able to discuss and write about the computer-controlled automation affects a workplace and society. Students will be able to discuss and write about the ramifications of society's dependence on computers. Students will be able to discuss and write about the advantages and disadvantages of changing workplace environments. Objective 3 - **Demonstrate knowledge of the right to privacy.** Students will be able to discuss and write about the computers that can compromise privacy. Students will be able to discuss and write about the knowledge of privacy laws.	viruses worms Trojans robust international copyright Informative Infrastructure Task Force (IITF) Intellectual Property trademark firewall typosquatting

▪ Objective 4 Demonstrate knowledge of computer, information and software security. a. Exhibit knowledge of copyright laws. b. Explain how computers could erroneously be used to compromise copyright laws. c. Give examples of ways to protect information on computer systems. d. Identify ways to protect against computer viruses.	Objective 4 - **Demonstrate knowledge of computer, information and software security.** Knowledge of copyright laws. Computers could erroneously be used to compromise copyright laws. Examples of ways to protect information on computer systems. Protect against computer viruses.	Students will be able to discuss and write about the responsibilities of people who control computer information. Objective 4 - **Demonstrate knowledge of computer, information and software security.** Students will be able to discuss and write about the knowledge of copyright laws. Students will be able to discuss and write about the computers that could erroneously be used to compromise copyright laws. Students will be able to discuss and write about the examples of ways to protect information on computer systems. Students will be able to discuss and write about the protection against computer viruses.