

**COURSE CSE1010: COMPUTER SCIENCE 1**

**Level:** Introductory

**Prerequisite:** None

**Description:** Students explore hardware, software and processes. This includes an introduction to the algorithm as a problem-solving tool, to programming languages in general and to the role of programming as a tool for implementing algorithms.

**Parameters:** Access to an appropriate computer work station, the Internet, a programming language/environment and associated support materials. It is recommended that the course be taught in tandem with one or more programming courses.

**Supporting Courses:** CSE1110: Structured Programming 1  
CSE1120: Structured Programming 2, and/or any  
Intermediate project course involving imperative programming

**Outcomes:** The student will:

- 1. identify and describe the nature, approaches and areas of interest of computer science**
  - 1.1 define and describe computer science with consideration of:
    - 1.1.1 the main goal of the discipline
    - 1.1.2 the use of algorithms
    - 1.1.3 computer systems used to test and/or implement algorithms
    - 1.1.4 the translation of algorithms through programming
  - 1.2 describe the general areas of interest of computer science including:
    - 1.2.1 the theory of computation
    - 1.2.2 algorithms and data structures
    - 1.2.3 programming methodology and languages
    - 1.2.4 computer elements and architecture
    - 1.2.5 human-machine and machine-machine interfacing
    - 1.2.6 automata
    - 1.2.7 artificial intelligence
    - 1.2.8 visual and auditory rendering
    - 1.2.9 general development of information technology applications
  - 1.3 compare and contrast computer science, computer engineering and information technology; e.g., theoretical versus applied, general versus specific, exploratory versus applicatory
  - 1.4 describe some of the misconceptions associated with computer science; e.g., synonymous with programming, reliant on solitary individuals for the bulk of its advances, relatively little real-world contact, the learning of various computer applications
  - 1.5 computer science's role in an information society
- 2. demonstrate an understanding of the nature, design and use of basic algorithms associated with problems involving the sequential inputting, processing and outputting of data**
  - 2.1 define algorithms and explain how they are used
  - 2.2 compare and contrast the "iterative and incremental" and "waterfall" models of software development

- 2.3 demonstrate the analysis and design stages of a Systems Development Life Cycle model using appropriate tools; e.g., flowcharts, pseudocode, input/processing/output (IPO) charting
- 2.4 demonstrate a number of core algorithms including:
  - 2.4.1 accumulation (keeping a running total)
  - 2.4.2 determining the mean
  - 2.4.3 determining minimums and maximums
- 3. explain and demonstrate the nature of structured programming**
  - 3.1 consider the rationale for structured programming
  - 3.2 consider GOTO-less programming
  - 3.3 consider three fundamental control structures—sequential, decision and iterative
- 4. explain and demonstrate an understanding of the nature, evolution, types and role of programming languages**
  - 4.1 describe how various programming languages have dealt with data representation; e.g., binary and hexadecimal systems, standard data types, data storage
  - 4.2 describe the nature of programming language, specifically that these languages:
    - 4.2.1 reflect a simplified version of natural language
    - 4.2.2 evolved in tandem with algorithms and hardware over a number of generations
    - 4.2.3 reflect the IPO data processing paradigm
  - 4.3 describe and demonstrate how programming languages are used in the coding stage of a Systems Development Life Cycle model by converting a representative set of algorithms into executable code
- 5. explain the nature, evolution and basic architecture of a von Neumann computer system**
  - 5.1 create a block diagram of a stereotypical von Neumann machine
  - 5.2 describe a number of typical devices associated with each block
  - 5.3 show the flow of data through the computer under the direction of a program
- 6. demonstrate basic competencies**
  - 6.1 demonstrate fundamental skills to:
    - 6.1.1 communicate
    - 6.1.2 manage information
    - 6.1.3 use numbers
    - 6.1.4 think and solve problems
  - 6.2 demonstrate personal management skills to:
    - 6.2.1 demonstrate positive attitudes and behaviours
    - 6.2.2 be responsible
    - 6.2.3 be adaptable
    - 6.2.4 learn continuously
    - 6.2.5 work safely
  - 6.3 demonstrate teamwork skills to:
    - 6.3.1 work with others
    - 6.3.2 participate in projects and tasks
- 7. make personal connections to the cluster content and processes to inform possible pathway choices**
  - 7.1 complete/update a personal inventory; e.g., interests, values, beliefs, resources, prior learning and experiences
  - 7.2 create a connection between a personal inventory and occupational choices



- 2.4.5 use literals and input commands, e.g., methods or operators, to provide data for processing
- 2.4.6 use assignment, arithmetical and concatenation and interpolation operators, where appropriate, to process data
- 2.4.7 use output commands; e.g., methods or operators, to display processed data
- 2.5 test the algorithm for failure or success with appropriate data
- 2.6 revise the algorithm, as required
- 3. analyze and compare the results of the program with the intent of the algorithm and modify as required**
  - 3.1 use appropriate test data and debugging techniques to track and correct errors including:
    - 3.1.1 run-time errors; e.g., compiler, linker, syntax
    - 3.1.2 logic errors
- 4. demonstrate basic competencies**
  - 4.1 demonstrate fundamental skills to:
    - 4.1.1 communicate
    - 4.1.2 manage information
    - 4.1.3 use numbers
    - 4.1.4 think and solve problems
  - 4.2 demonstrate personal management skills to:
    - 4.2.1 demonstrate positive attitudes and behaviours
    - 4.2.2 be responsible
    - 4.2.3 be adaptable
    - 4.2.4 learn continuously
    - 4.2.5 work safely
  - 4.3 demonstrate teamwork skills to:
    - 4.3.1 work with others
    - 4.3.2 participate in projects and tasks
- 5. make personal connections to the cluster content and processes to inform possible pathway choices**
  - 5.1 complete/update a personal inventory; e.g., interests, values, beliefs, resources, prior learning and experiences
  - 5.2 create a connection between a personal inventory and occupational choices

## **COURSE CSE1120:    STRUCTURED PROGRAMMING 2**

**Level:**                    Introductory

**Prerequisite:**         CSE1110: Structured Programming 1

**Description:**         Students work with structured programming constructs by adding the selection and iteration program control flow mechanisms to their programming repertoire. They write structured algorithms and programs that use blocks to introduce an element of modularity into their programming practice.

**Parameters:**         Access to appropriate computer equipment, software, to the Internet and support materials. Specifically, students must have access to a programming environment that encourages structured programming.

**Supporting Courses:**   CSE1010: Computer Science 1, or any Intermediate project course involving imperative programming

**Outcomes:**            The student will:

- 1. demonstrate basic structured programming skills by writing algorithms to solve problems involving selection (decision making) and iteration (repetition)**
  - 1.1 analyze a problem and determine if it can be solved using an algorithm that employs an input/processing/output (IPO) approach
  - 1.2 determine if there is more than one IPO module present
  - 1.3 decompose the problem into its respective modules and identify the IPO components of each module
  - 1.4 identify what data is already available to the programmer and what must be inputted by the end user and organize into appropriate block or blocks using the appropriate program control structures
  - 1.5 identify the processing requirements and organize into appropriate blocks using the appropriate program control structures
  - 1.6 incorporate basic algorithmic idioms as required; e.g., accumulation, determining maximum or minimum values
  - 1.7 identify the output requirements and organize into appropriate blocks using the appropriate program control structures
  - 1.8 order components into an appropriate sequence where processing occurs only when all required data for a module is available and output occurs only after appropriate processing has occurred
  - 1.9 write the algorithm in an acceptable format; e.g., pseudocode, a structured chart
- 2. translate algorithms into source code, convert the source code into machine executable form, execute and debug, as required**
  - 2.1 maintain the IPO structure of the algorithm
  - 2.2 use appropriate internal and external documentation
  - 2.3 use appropriate basic (primitive) data types such as integers, real numbers, characters, strings, and Boolean values
  - 2.4 use appropriate variables and constants to hold data
  - 2.5 use literals and input commands, e.g., methods or operators, to provide data for processing

- 2.6 use assignment, arithmetical, relational, Boolean, and concatenation and interpolation operators, where appropriate, to process data
- 2.7 use basic processing idioms as required; e.g., accumulation, determining maximum or minimum values
- 2.8 use appropriate selection and iteration structures to avoid unconditional branching or exiting from the interior of a block including:
  - 2.8.1 nested conditional blocks
  - 2.8.2 nested iterative blocks
- 2.9 use output commands, e.g., methods or operators, to display processed data in an appropriately formatted form
- 3. analyze and compare the results of the program with the intent of the algorithm and modify, as required**
  - 3.1 use appropriate test data and debugging techniques to track and correct errors including:
    - 3.1.1 run-time errors; e.g., compiler, linker, syntax
    - 3.1.2 logic errors
- 4. demonstrate basic competencies**
  - 4.1 demonstrate fundamental skills to:
    - 4.1.1 communicate
    - 4.1.2 manage information
    - 4.1.3 use numbers
    - 4.1.4 think and solve problems
  - 4.2 demonstrate personal management skills to:
    - 4.2.1 demonstrate positive attitudes and behaviours
    - 4.2.2 be responsible
    - 4.2.3 be adaptable
    - 4.2.4 learn continuously
    - 4.2.5 work safely
  - 4.3 demonstrate teamwork skills to:
    - 4.3.1 work with others
    - 4.3.2 participate in projects and tasks
- 5. make personal connections to the cluster content and processes to inform possible pathway choices**
  - 5.1 complete/update a personal inventory; e.g., interests, values, beliefs, resources, prior learning and experiences
  - 5.2 create a connection between a personal inventory and occupational choices

**COURSE CSE1210: CLIENT-SIDE SCRIPTING 1**

**Level:** Introductory

**Prerequisite:** None

**Description:** Students are introduced to Internet computing through the use of one or more Web-specific markup languages. As part of this process, students learn how the Web uses markup languages to provide a client-side approach to display static information. Students also learn how to analyze, modify, write and debug algorithms and documents that use a markup language.

**Parameters:** Access to appropriate computer equipment, software, support materials and the Internet. More specifically, students must have the tools they will require to design, write and debug markup language-based hypermedia documents.

**Supporting Courses:** CSE1010: Computer Science 1  
CSE1110: Structured Programming 1

**Outcomes:** The student will:

**1. demonstrate an understanding of the general architecture of the Internet as it pertains to client-side scripting**

- 1.1 explain and demonstrate the client/server nature of the Internet including:
  - 1.1.1 describe the internetworked nature of the Internet
  - 1.1.2 illustrate the client/server relationship that exists among the work stations, stub networks, mid-level networks, regional networks and the backbone that makes up the Internet
  - 1.1.3 describe and illustrate how servers, routers, switches, work stations and other hardware components are used to provide the physical matrix required for client/server relationships
  - 1.1.4 describe and illustrate the Internet Protocol Suite (TCP/IP) model of networking, in general terms, outlining how this protocol provides the data transfer mechanism required to establish client/server relationships
  - 1.1.5 illustrate the client/server relationships set up when a user makes a request for an Internet service and that request is carried out
  - 1.1.6 describe at least three examples of Internet services that rely on client/server relationships
- 1.2 explain and demonstrate the client/server nature of the Web including:
  - 1.2.1 describe the hypertext-based nature of the Web
  - 1.2.2 describe the relationship between the Web and the Internet as a whole
  - 1.2.3 explain why the Web can be thought of as a network of hyperlinked documents
  - 1.2.4 describe and illustrate the client/server relationship that allows user agents to interact with the origin servers that make up the information repository components of the Web
  - 1.2.5 describe how the Hypertext Transfer Protocol (HTTP) is used to facilitate client/server interaction
  - 1.2.6 describe and illustrate the general flow of information through the Internet when a user agent uses a Web browser to interact with origin servers
  - 1.2.7 explain how HTTP is used to protect the transmission of data through the Web
  - 1.2.8 describe and illustrate the development of the Web in general terms using the Web 1.0, Web 2.0 and Web 3.0 generational paradigms
  - 1.2.9 compare and contrast the Web 1.0, Web 2.0 and Web 3.0 stages of development

- 2. demonstrate an understanding of the general nature and purpose of Internet-oriented markup languages**
  - 2.1 describe the role markup languages play in the Web
  - 2.2 compare and contrast markup and scripting languages
  - 2.3 describe and illustrate the development of Internet-oriented markup languages in general terms including:
    - 2.3.1 explain the relationship between Standard Generalized Markup Language (SGML), Extensible Markup Language (XML), Hypertext Markup Language (HTML), Extensible Hypertext Markup Language (XHTML) and Dynamic Hypertext Markup Language (DHTML)
    - 2.3.2 explain at least two specialized Internet markup languages
- 3. design, write and debug code using an appropriate Internet markup language**
  - 3.1 demonstrate the ability to use an appropriate markup language coding environment
  - 3.2 use appropriate techniques to design a markup language document including:
    - 3.2.1 determine and outline the intent of the document
    - 3.2.2 organize the document into appropriate subsections or pages
    - 3.2.3 describe the content to be carried on each page
    - 3.2.4 illustrate how the content is to be displayed
    - 3.2.5 identify the locations of the required anchors and links
  - 3.3 translate design documents into hypertext documents using code elements such as tags, attributes and hyperlinks to:
    - 3.3.1 mark off the various parts of the document
    - 3.3.2 display text and visual data in a variety of formats
    - 3.3.3 create specialized formats such as lists, tables and frames
    - 3.3.4 create both textual and image-based hyperlinks; e.g., both single images and mapped images
  - 3.4 compare the results of the script with the intent of the design document and modify, as required, including:
    - 3.4.1 use appropriate debugging techniques to compare the original design with the implemented document
    - 3.4.2 make changes, as required, to either the design and/or the document to bring both in line with the original intent
- 4. demonstrate basic competencies**
  - 4.1 demonstrate fundamental skills to:
    - 4.1.1 communicate
    - 4.1.2 manage information
    - 4.1.3 use numbers
    - 4.1.4 think and solve problems
  - 4.2 demonstrate personal management skills to:
    - 4.2.1 demonstrate positive attitudes and behaviours
    - 4.2.2 be responsible
    - 4.2.3 be adaptable
    - 4.2.4 learn continuously
    - 4.2.5 work safely
  - 4.3 demonstrate teamwork skills to:
    - 4.3.1 work with others
    - 4.3.2 participate in projects and tasks
- 5. make personal connections to the cluster content and processes to inform possible pathway choices**
  - 5.1 complete/update a personal inventory; e.g., interests, values, beliefs, resources, prior learning and experiences
  - 5.2 create a connection between a personal inventory and occupational choices

## **COURSE CSE1220: CLIENT-SIDE SCRIPTING 2**

**Level:** Introductory

**Prerequisite:** None

**Note:** CSE1210: Client-side Scripting 1 or an equivalent course dealing with markup scripting is strongly recommended

**Description:** Students deepen their understanding of Internet computing by using more advanced markup language techniques and by being introduced to one or more Web-specific scripting languages. As part of this process, students learn how the Web uses these resources as a means of displaying dynamic client-side information. Students learn how to analyze, modify, write and debug algorithms and scripts that use structured programming approaches.

**Parameters:** Access to appropriate computer equipment, software, support materials and the Internet. More specifically, students must have the tools they will require to design, write and debug hypermedia documents and Internet scripts.

**Supporting Courses:** CSE1210: Client-side Scripting 1  
CSE1110: Structured Programming 1  
CSE1120: Structured Programming 2

**Outcomes:** The student will:

### **1. compare and contrast static and dynamic client-side scripting**

- 1.1 describe and illustrate the main differences, from the user's perspective, between dynamic and static client-side Web sites
- 1.2 describe and illustrate the main differences between how dynamic and static client-side sites are implemented by the Web
- 1.3 describe and illustrate the main advantages and disadvantages of dynamic and static client-side sites

### **2. design, write and debug scripts that use advanced markup language approaches to provide some aspects of dynamic client-side site construction**

- 2.1 describe how markup languages can be used to provide a limited amount of client-side dynamic display through the use of forms and style sheets
- 2.2 use appropriate techniques to design a hypertext document that employs forms and style sheets including:
  - 2.2.1 determine and outline the intent of the document
  - 2.2.2 use appropriate problem decomposition techniques to organize the document into smaller components or pages and to identify locations of any required anchors and links
  - 2.2.3 determine the content to be carried on each page and how that content is to be displayed
  - 2.2.4 identify what role style sheets have in controlling page display
  - 2.2.5 determine what role forms have in managing input and subsequent interaction
- 2.3 write and debug scripts that translate design documents employing forms and style sheets into client-side sites by:
  - 2.3.1 identifying and using appropriate techniques for separating content and presentation through the use of inline and/or embedded style sheets
  - 2.3.2 identifying and using appropriate techniques for soliciting user input through the use of forms

- 2.3.3 using appropriate techniques to determine if the script will achieve the original intent
- 2.3.4 using appropriate internal and external documentation
- 3. describe the general nature and purpose of Internet-oriented scripting languages**
  - 3.1 describe the role scripting languages play in the creation of Web sites
  - 3.2 compare and contrast scripting languages with markup languages and with general purpose programming languages
  - 3.3 describe and illustrate the development of Internet-oriented scripting languages in general terms
  - 3.4 describe and illustrate at least two specialized Internet-oriented scripting languages; e.g., Javascript, PERL
- 4. design, write and debug scripts that use structured programming approaches with an appropriate Internet-oriented scripting language**
  - 4.1 demonstrate the ability to use an appropriate scripting language coding environment
  - 4.2 outline the intent of the script and determine if the intent can be realized using structured programming approaches
  - 4.3 write algorithms that use structured programming approaches to realize the intent of the script including:
    - 4.3.1 use appropriate problem decomposition techniques to break the problem into smaller components
    - 4.3.2 identify the input, processing and output requirements of each component
    - 4.3.3 further decompose each component into smaller blocks, as required, using the appropriate structures to control program flow
    - 4.3.4 write the algorithm in an acceptable format
    - 4.3.5 use appropriate techniques to determine if the algorithm will achieve the original intent
  - 4.4 translate the algorithm into a script using structured programming approaches by:
    - 4.4.1 maintaining the structure of the algorithm
    - 4.4.2 using appropriate basic or primitive data types
    - 4.4.3 using appropriate variables and constants, as required, to hold data
    - 4.4.4 using literals and input commands to provide data for processing
    - 4.4.5 using assignment, arithmetical, relational, Boolean and, where available, concatenation and string construction operators to process data
    - 4.4.6 using basic processing idioms such as accumulation, determining maximum or minimum values, as required
    - 4.4.7 using appropriate selection and iteration structures such as conditional and iterative blocks
    - 4.4.8 using output commands to display processed data in an appropriately formatted form
    - 4.4.9 using appropriate internal and external documentation
  - 4.5 execute the script tracking and eradicating errors including:
    - 4.5.1 embed the script in an appropriate markup document
    - 4.5.2 eliminate run-time and logic errors
  - 4.6 compare the results of the script's execution with the intents of the algorithm and modify, as required
- 5. demonstrate basic competencies**
  - 5.1 demonstrate fundamental skills to:
    - 5.1.1 communicate
    - 5.1.2 manage information
    - 5.1.3 use numbers
    - 5.1.4 think and solve problems

- 5.2 demonstrate personal management skills to:
  - 5.2.1 demonstrate positive attitudes and behaviours
  - 5.2.2 be responsible
  - 5.2.3 be adaptable
  - 5.2.4 learn continuously
  - 5.2.5 work safely
- 5.3 demonstrate teamwork skills to:
  - 5.3.1 work with others
  - 5.3.2 participate in projects and tasks
- 6. make personal connections to the cluster content and processes to inform possible pathway choices**
  - 6.1 complete/update a personal inventory; e.g., interests, values, beliefs, resources, prior learning and experiences
  - 6.2 create a connection between a personal inventory and occupational choices





- 2. use a general understanding of robotics to analyze a robot operating within its environment**
  - 2.1 describe its architecture
  - 2.2 indicate how it displays “agency” or autonomous action
  - 2.3 categorize its control system
  - 2.4 describe its capabilities
  - 2.5 describe its relationship with its environment
  - 2.6 identify at least one task that the robot should be able to accomplish within its environment
  - 2.7 explain how either the robot and/or its environment could be modified to increase the number and type of tasks it could accomplish
- 3. design a robotics system consisting of at least one robot, associated control systems and environment capable of carrying out a simple set of predetermined tasks**
  - 3.1 identify the general tasks the robot will be required to carry out by:
    - 3.1.1 breaking tasks into simpler tasks and continuing the process until the tasks are reduced to primitives
    - 3.1.2 establishing the task sequence and creating a representation of those tasks
  - 3.2 describe and illustrate the environment in which the robot will be required to operate by:
    - 3.2.1 identifying elements of the environment that the robot will be able to manipulate
    - 3.2.2 identifying manipulatable elements that will act as task resources and task obstacles
    - 3.2.3 identifying variations in the environment that the robot will be able to detect; e.g., light, colour, sound
    - 3.2.4 setting the outer limits of the environment
    - 3.2.5 determining the location and type of internal barriers in the environment
    - 3.2.6 incorporating safety elements into the environment that will protect both the humans and robots operating in the environment
  - 3.3 identify the capabilities the robot will require to carry out set tasks including:
    - 3.3.1 sensing requirements
    - 3.3.2 mobility requirements
    - 3.3.3 manipulation requirements
    - 3.3.4 power requirements
    - 3.3.5 processing requirements; e.g., both calculation and data storage
  - 3.4 determine the control approach to be used to:
    - 3.4.1 determine if the robot has the capacity for autonomous operation
    - 3.4.2 determine what level of operator control will be required if the robot is not fully autonomous
  - 3.5 design the robot using the tasks to be accomplished, proposed environment, required capabilities, and control approach as parameters to determine what:
    - 3.5.1 type of kinematic chain or body will be required to provide a platform for the other components
    - 3.5.2 actuators and end effectors will be required and how they will be mounted on the body
    - 3.5.3 sensors will be required and how they will be mounted on the body
    - 3.5.4 control components will be required and how they will be mounted on the body
    - 3.5.5 power or energy components will be required and how they will be mounted on the body
  - 3.6 check your design for congruency against the task list to be accomplished and with the proposed environmental specifications
  - 3.7 modify the design, as required
  - 3.8 carry out the design process sequentially using a top-down approach and employing stepwise refinement
- 4. use an iterative process to build the environment, robot and controlling mechanism called for by the design**
  - 4.1 construct that portion of the environment required for the first task or tasks in the task sequence
  - 4.2 assemble as much of the robot, as is required, to accomplish the task or tasks

- 4.3 write algorithms that use structured programming approaches to accomplish the task or set of tasks including:
    - 4.3.1 use appropriate problem decomposition techniques to break the task into subtasks
    - 4.3.2 identify the sense, plan and action requirements of each subtask
    - 4.3.3 further decompose each subtask into smaller blocks, as required, using the appropriate structures to control program flow
    - 4.3.4 write the algorithm in an approved format such as a structured chart or pseudocode
    - 4.3.5 use appropriate techniques to determine if the algorithm will achieve the original intent
  - 4.4 use an RCL capable of writing structured code to translate the algorithm for the set of tasks into a program including:
    - 4.4.1 maintain the structure of the algorithm
    - 4.4.2 use appropriate basic or primitive data types and variable and constant names, as required, to hold data
    - 4.4.3 use literals and input commands to accept data from sensors to provide data for processing
    - 4.4.4 use operators and basic processing idioms as required; e.g., accumulation, determination of maximum or minimum values
    - 4.4.5 use appropriate selection and iteration structure; e.g., conditional, iterative blocks
    - 4.4.6 use output commands to display processed data to the operator as well as activate actuators; e.g., motors, grippers
    - 4.4.7 document appropriately
  - 4.5 load and execute the program tracking and eradicating errors by:
    - 4.5.1 testing each of the physical subsystems of the robot(s) to eliminate engineering errors
    - 4.5.2 testing the robot(s) within the appropriate section of the environment to confirm that the robot is interacting with the environment as called for by the algorithm
    - 4.5.3 using self-test code and check points, as well as observation, to eliminate run-time and internal logic errors
    - 4.5.4 comparing the robot's actions with the intent of the algorithm
    - 4.5.5 modifying the original task list, environment, algorithm and/or program, as required
  - 4.6 participate in interim critiques throughout the iterative process; e.g., planning, analysis, design, testing, evaluation
- 5. demonstrate basic competencies**
- 5.1 demonstrate fundamental skills to:
    - 5.1.1 communicate
    - 5.1.2 manage information
    - 5.1.3 use numbers
    - 5.1.4 think and solve problems
  - 5.2 demonstrate personal management skills to:
    - 5.2.1 demonstrate positive attitudes and behaviours
    - 5.2.2 be responsible
    - 5.2.3 be adaptable
    - 5.2.4 learn continuously
    - 5.2.5 work safely
  - 5.3 demonstrate teamwork skills to:
    - 5.3.1 work with others
    - 5.3.2 participate in projects and tasks
- 6. make personal connections to the cluster content and processes to inform possible pathway choices**
- 6.1 complete/update a personal inventory; e.g., interests, values, beliefs, resources, prior learning and experiences
  - 6.2 create a connection between a personal inventory and occupational choices



## **COURSE CSE1910: CSE PROJECT A**

**Level:** Introductory

**Prerequisite:** None

**Description:** Students develop project design and management skills to extend and enhance competencies and skills in other CTS courses through contexts that are personally relevant.

**Parameters:** Introductory project courses must connect with a minimum of two CTS courses, one of which must be at the introductory level and be in the same occupational area as the project course. The other CTS course(s) can be either at the same level or at the intermediate level from any occupational area.

Project courses cannot be connected to other project courses or practicum courses.

**All projects and/or performances, whether teacher- or student-led, must include a course outline or student proposal.**

### **Outcomes:**

The teacher/student will:

- 1. identify the connection between this project course and two or more CTS courses**
  - 1.1 identify the outcome(s) from each identified CTS course that support the project and/or performance deliverables
  - 1.2 explain how these outcomes are being connected to the project and/or performance deliverables
- 2. propose the project and/or performance**
  - 2.1 identify the project and/or performance by:
    - 2.1.1 preparing a plan
    - 2.1.2 clarifying the purposes
    - 2.1.3 defining the deliverables
    - 2.1.4 specifying time lines
    - 2.1.5 explaining terminology, tools and processes
    - 2.1.6 defining resources; e.g., materials, costs, staffing
  - 2.2 identify and comply with all related health and safety standards
  - 2.3 define assessment standards (indicators for success)
  - 2.4 present the proposal and obtain necessary approvals

The student will:

- 3. meet goals as defined within the plan**
  - 3.1 complete the project and/or performance as outlined
  - 3.2 monitor the project and/or performance and make necessary adjustments
  - 3.3 present the project and/or performance, indicating the:
    - 3.3.1 outcomes attained
    - 3.3.2 relationship of outcomes to goals originally set

- 3.4 evaluate the project and/or performance, indicating the:
  - 3.4.1 processes and strategies used
  - 3.4.2 recommendations on how the project and/or performance could have been improved
- 4. demonstrate basic competencies**
  - 4.1 demonstrate fundamental skills to:
    - 4.1.1 communicate
    - 4.1.2 manage information
    - 4.1.3 use numbers
    - 4.1.4 think and solve problems
  - 4.2 demonstrate personal management skills to:
    - 4.2.1 demonstrate positive attitudes and behaviours
    - 4.2.2 be responsible
    - 4.2.3 be adaptable
    - 4.2.4 learn continuously
    - 4.2.5 work safely
  - 4.3 demonstrate teamwork skills to:
    - 4.3.1 work with others
    - 4.3.2 participate in projects and tasks
- 5. make personal connections to the cluster content and processes to inform possible pathway choices**
  - 5.1 complete/update a personal inventory; e.g., interests, values, beliefs, resources, prior learning and experiences
  - 5.2 create a connection between a personal inventory and occupational choices