Computer Science 110

September 5, 2019

New Nouveau Brunswick

2019
Department of Education and Early Childhood Development
Curriculum Branch

Acknowledgments

The Department of Education and Early Childhood Development of New Brunswick gratefully acknowledges the contributions of the following groups and individuals toward the development of the New Brunswick Computer Science 110 Curriculum Document:

- Computer Science 110 Curriculum Development Advisory Committee of:
  - Nicholas Fullerton (ASD-North)
  - Catherine Duffy (ASD-East)
  - Patrick Bidlake (ASD-West / Online Teacher)
  - Craig Duplessie (ASD-North)
  - Brian Gray, Learning Specialist for Skilled Trades and Technology (EECD)
  - Mike Cusack, Learning Specialist for Distance and Online Learning (EECD)
  - Graham Rich, Learning Specialist for Information and Communication Technology (EECD)

A special thanks to the following individuals for attending our four-day training session (July 2016) and also for their leadership in sharing their expertise, their learning materials, and their vision:

- Computer Science 110 Sharing and Support (continuing since 2016):
  - Catherine Duffy (ASD-East)
  - Patrick Bidlake (ASD-West / Online Teacher)
  - Andrew Colwell (ASD-South)
- Continuous Promotion of Computer Science and Cybersecurity in New Brunswick
  - Jamie Rees (NB Power)

Table of Contents

1. Introduction

1.1 Mission and Vision of Educational System

The New Brunswick Department of Education and Early Childhood Development is dedicated to providing the best public education system possible, wherein all students have a chance to achieve their academic best. The mission statement for New Brunswick schools is:

> *Each student will develop the attributes needed to be a lifelong learner, to achieve personal fulfillment and to contribute to a productive, just and democratic society.*

1.2 New Brunswick Global Competencies

New Brunswick Global Competencies provide a consistent vision for the development of a coherent and relevant curriculum. The statements offer students clear goals and a powerful rationale for school work. They help ensure that provincial education systems' missions are met by design and intention. The New Brunswick Global Competencies statements are supported by curriculum outcomes.

New Brunswick Global Competencies are statements describing the knowledge, skills and attitudes expected of all students who graduate high school. Achievement of the New Brunswick Global Competencies prepares students to continue to learn throughout their lives. These Competencies describe expectations not in terms of individual school subjects but in terms of knowledge, skills and attitudes developed throughout the curriculum. They confirm that students need to make connections and develop abilities across subject boundaries if they are to be ready to meet the shifting and ongoing demands of life, work and study today and in the future.

See Appendix 6.1.

2. Pedagogical Components

2.1 Pedagogical Guidelines

Diverse Cultural Perspectives

It is important for teachers to recognize and honour the variety of cultures and experiences from which students are approaching their education and the world. It is also important for teachers to recognize their own biases and be careful not to assume levels of physical, social or academic competencies based on gender, culture, or socio-economic status.

Each student's culture will be unique, influenced by their community and family values, beliefs, and ways of viewing the world. Traditional aboriginal culture views the world in a much more holistic way than the dominant culture. Disciplines are taught as connected to one another in a practical context, and learning takes place through active participation, oral communication and experiences. Immigrant students may also be a source of alternate world views and cultural understandings. Cultural variation may arise from the differences between urban, rural and isolated communities. It may also arise from the different value that families may place on academics or athletics, books or media, theoretical or practical skills, or on community and church. Providing a variety of teaching and assessment strategies to build on this diversity will provide an opportunity to enrich learning experiences for all students.

Universal Design for Learning

The curriculum has been created to support the design of learning environments and lesson plans that meet the needs of all learners. Specific examples to support Universal Design for Learning for this curriculum can be found in the appendices. The **Planning for All Learners Framework** will guide and inspire daily planning.

See Appendix 6.2

English as an Additional Language Curriculum

Being the only official bilingual province, New Brunswick offers the opportunity for students to be educated in English and/or French through our public education system. The EECD provides leadership from K-12 to assist educators and many stakeholders in supporting newcomers to New Brunswick. English language learners have opportunities to receive a range of instructional support to improve their English language proficiency through an inclusive learning environment. EECD, in partnership with the educational and wider communities offer a solid, quality education to families with school-aged children.

2.2 Pedagogical Guidelines

Assessment Practices

Assessment is the systematic gathering of information about what students know and are able to do. Student performance is assessed using the information collected during the evaluation process. Teachers use their professional skills, insight, knowledge, and specific criteria that they establish to make judgments about student performance in relation to learning outcomes. Students are also encouraged to monitor their own progress through self-assessment strategies, such as goal setting and rubrics.

Research indicates that students benefit most when assessment is regular and ongoing and is used in the promotion of learning (Stiggins, 2008). This is often referred to as formative assessment. Evaluation is less effective if it is simply used at the end of a period of learning to determine a mark (summative evaluation).

Summative evaluation is usually required in the form of an overall mark for a course of study, and rubrics are recommended for this task. Sample rubrics templates are referenced in this document, acknowledging teachers may have alternative measures they will apply to evaluate student progress.

Some examples of current assessment practices include:

• Questioning	• Projects and Investigations
• Observation	• Checklists/Rubrics
• Conferences	• Responses to texts/activities
• Demonstrations	• Reflective Journals
• Presentations	• Self and peer assessment
• Role plays	• Career Portfolios
• Technology Applications	• Projects and Investigations

Formative Assessment

Research indicates that students benefit most when assessment is ongoing and is used in the promotion of learning (Stiggins, 2008). Formative assessment is a teaching and learning process that is frequent and interactive. A key component of formative assessment is providing ongoing feedback to learners on their understanding and progress. Throughout the process adjustments are made to teaching and learning.

Students should be encouraged to monitor their own progress through goal setting, co-constructing criteria and other self-and peer-assessment strategies. As students become more involved in the assessment process, they are more engaged and motivated in their learning.

Additional details can be found in the Formative Assessment document.

Summative Assessment

Summative evaluation is used to inform the overall achievement for a reporting period for a course of study. Rubrics are recommended to assist in this process. Sample rubrics templates are referenced in this document, acknowledging teachers may have alternative measures they will apply to evaluate student progress.

For further reading in assessment and evaluation, visit the Department of Education and Early Childhood Development's Assessment and Evaluation site here.

Cross Curricular Literacy

Literacy occurs across learning contexts and within all subject areas. Opportunities to speak and listen, read and view, and write and represent are present every day -in and out of school.

3. Subject Specific Guidelines

3.1 Rationale

The field of computer science has evolved significantly since the previous Computer Science 110 curriculum (circa 1997). This curriculum refresh seeks to better prepare secondary students for post-secondary and industry opportunities, as based on input that representatives consider key in this field.

During this program of study, students will be challenged through the lens of project-based learning. The curriculum outcomes demonstrate the commitment to New Brunswick's implementation of 'Global Competencies'. Through the collaborative projects in this course, students work toward these outcomes in learning activities that are meaningful and focused on the student. These Global Competencies should not be interpreted as instructional pathways but rather expectations to be met simultaneously with the skills and knowledge required in this course.

The required knowledge and specific skills shown in the outcomes have been limited in quantity, to facilitate students' deeper investigation and application of the curriculum topics, while also providing flexibility for instructors as the field of computer science has been demonstrated to shift over time, as seen in languages, components, applications and mediums; of late, computer science has experienced unprecedented shifts, as the tools become more powerful and more accessible to students, including such tools as big data, chat bots, ransomware, machine learning, neural networks, artificial intelligence and virtual assistants (e.g., Siri and Alexa). We expect these shifts to continue, and this curriculum provides teachers the ability to shift alongside new technologies, giving New Brunswick students the best possible foundation for future education and careers.

The primary purpose of this course is that students will gain a strong foundation while demonstrating operational skills, using computational thinking, incorporating team work, meeting challenges, solving problems, and being resilient and resourceful. All of these are what we heard from our industry focus group, and what we have continued to hear from educational experts as this curriculum refresh has taken shape. In addition to computer programming and networking skills, we hope that teachers will instill in our students an understanding of how computer science and cybersecurity are closely linked together.

3.2 Course Description

The Computer Science 110 (CS110) course will inspire students through the experiential learning of the fundamentals and possibilities of computer science. In CS110 students will be actively engaged in the design, development and evaluation of computer science projects. The intent of this program of study is to have students coding from day one and maintaining a high level of engagement throughout the course through a commitment to problem and project-based learning. To achieve a high level of student engagement, teachers use feedback-loop of instruction, hands-on learning, and assessment. See example below:

- Present one fundamental of computer coding (e.g. conditionals)
- What is the *computational thinking* behind this fundamental? (e.g. how "if" works)
- What *planning* surrounds this fundamental concept? (e.g. how "if" looks on a flowchart)
- What *syntax* does this fundamental require (e.g. "if (____) { }")
- How is this *concept* used in my project? (e.g. where, why and how to include the "if" code)
- Has the student demonstrated proper use of the concept? (e.g. is "if" used correctly)

This curriculum committee recognizes that students will be entering this course with differing levels of experience and competence, so teachers are encouraged to prepare multiple learning pathways and/or methodologies for students. In our inclusive classes, it is rare that a "one size fits all" approach will be successful in attracting students to continue further with computer science; therefore, teachers need to be prepared, and depending on students' prior coding experience, the teacher's starting point may need to be adjusted. Some potential student paths are suggested below.

Level of coding exposure and projected Computer Science 110 and Computer Science 120 pathway:

High level	**Medium level**	**Low level**
MSTE Visual Coding May include text-based coding	MSTE None	MSTE None
BBT Review visual coding Text-based coding	BBT Visual coding	BBT None
CS 110 Advanced text-based coding Additional language(s)	CS 110 Mostly text-based coding	CS 110 Mix of visual and text-based coding
CS 120 Advanced text-based coding and additional languages	CS 120 Advanced text-based coding May include additional language(s)	CS 120 Advanced text-based coding

By the end of this course, students will have an awareness, understanding, and experience with computer science, especially in the context of computer programming (i.e., coding) and working with operating systems, networks, and hands-on learning tools.

3.3 Curriculum Organizers and Outcomes

Organizers

Computer Science 110 curriculum has been developed with digital literacy in mind, including inquiry, problem solving and decision making. Inquiry also involves empathy and understanding the challenges that humans are facing. Problem solving involves understanding the human challenges and brainstorming solutions based on a thorough understanding of the people facing the challenge and their expectations for any solution. Decision making involves both solution selection, project management, project testing and quality assurance, as well as evaluation. Decision making also involves selecting a development model (waterfall or agile) and understanding the advantages, disadvantages and consequences over the full lifespan of the project, and over the lifespan of professional projects that they may undertake later if they choose a career involving computer science and coding.

Outcomes

The New Brunswick Curriculum is stated in terms of general curriculum outcomes, specific curriculum outcomes, and achievement indicators.

General Curriculum Outcomes (GCO) are overarching statements about what students are expected to learn in each strand/sub-strand. The general curriculum outcome for each strand/sub-strand is the same throughout the grades.

Specific Curriculum Outcomes (SCO) are statements that identify specific concepts and related skills underpinned by the understanding and knowledge attained by students as required for a given grade.

By the end of Computer Science 110, students will:

GCO 1	**Demonstrate communication and operational skills specific to computer science.**
SCO 1.1	Students will persevere and demonstrate resourcefulness when challenges arise during a project.
SCO 1.2	Students will articulate challenges and hypothesize solutions to complete projects.
SCO 1.3	Students will use team-based project management strategies during collaborative efforts.
SCO 1.4	Students will apply the fundamentals of digital technology in relation to coding and computer science.
GCO 2	**Use computational thinking skills to analyze challenges and to create and evaluate solutions.**
SCO 2.1	Students will decompose a larger challenge into smaller manageable challenges.
SCO 2.2	Students will create repeatable solutions to manageable challenges.
SCO 2.3	Students will represent, collect and manage data to accomplish a task.
SCO 2.4	Students will create and use algorithms to organize and analyze data and information.
SCO 2.5	Students will execute a solution and evaluate a solution's validity, efficiency, and effectiveness.
GCO 3	**Demonstrate skills and analyze computer science artifacts.**
SCO 3.1	Students will analyze data needs and create artifacts that use data input, output and variables.
SCO 3.2	Students will analyze opportunities for modular code and create artifacts that use loops and other techniques that reuse code.
SCO 3.3	Students will analyze branching needs and create artifacts that use conditional statements.
SCO 3.4	Students will analyze opportunities for abstraction and create artifacts that demonstrate abstraction in code.
SCO 3.5	Students will create and analyze artifacts with understandable code, helpful names, and efficient comments and accompanying documentation.

4. Curriculum Outcomes

GCO 1	**Students will demonstrate communication and operational skills specific to computer science.**
SCO 1.1	**Students will persevere and demonstrate resourcefulness when challenges arise during a project.**

Concepts and Content	**Achievement Indicators**
Resourcefulness: • in the face of challenges, problems, errors and misleading information • to brainstorm, research and discuss possible causes and potential solutions Perseverance: • in the face of complex, ambiguous, non-trivial and difficult-to-grasp challenges • to keep focus on the immediate task and on the larger goal • to keep working and not back down, even if the task is more difficult than ever seen before to stay working on a task until it is solved or completed	I can be resourceful when brainstorming, researching, analyzing, and discussing potential solutions to challenges, problems and errors. I can continue to be resourceful in the face of ongoing challenges. I can persevere in my work that is complex, ambiguous, non-trivial, or difficult to grasp. I can stay focused on a task with multiple potential solutions and keep in mind both the task and the larger goal. I can keep working on a task, even though it is more difficult than any I've seen before. I can work on a project with complex tasks, whether alone or in a group, and I can see the project through to completion. I can find and use online coding resources and community-based support. I can critically evaluate proposed solutions and alternative approaches to solving a problem or responding to a challenge. I can debug my code iteratively.
Teacher Resources: https://nbed.sharepoint.com/sites/ComputerScience1102/	

SCO 1.2	**Students will articulate challenges and hypothesize solutions to complete projects.**
Concepts and Content	**Achievement Indicators**
Articulate challenges: • that may involve people and/or technology • that may involve technical vocabulary Hypothesize solutions: • that may be involved in completing projects and resolving coding challenges • to determine possible strategies and/or solutions to respond to a challenge or problem • to plan a coding project, including purpose, people, technology, pseudocode and documentation • to plan for secure code (proactively mitigating cybersecurity risks involving people and technology)	I can explain coding challenges that may involve people, technology, or both. I can analyze coding challenges that may involve people, technology, or both. I can be presented with a situation, challenge, or problem and I can create and maintain a list of potential solutions involving coding. I can identify useful Internet resources for coding challenges. I can hypothesize solutions based on the brainstorming, researching, analyzing,and discussing possible causes and potential solutions to coding challenges. I can use coding vocabulary effectively and I know that 'coding' and 'computer programming' can be used interchangeably. I can analyze information (e.g., device images, news article) about a current coding challenges and make a hypothesis about: • the types of devices potentially used in a solution; • what components would involve coding; • what actions would result from the coding; and • what improvements or additional challenges might still need to be addressed.
Teacher Resources: https://nbed.sharepoint.com/sites/ComputerScience1102/	

SCO 1.3	**Students will use team-based project management strategies during collaborative efforts.**
Concepts and Content	**Achievement Indicators**
Collaborate: • communicate effectively within a team • collaborate online and in-person on team-based tasks • to develop a Client-Programmer relationship through communication and collaboration • use at least one online collaboration platform for programming Project Management: • identify the roles in effective teams • identify and compare project management strategies • adopt and adapt roles and strategies for a specific group with a specific task • delegate tasks - and receive delegated tasks – that enable a successful coding project • determine group norms including methods for decision-making such as consensus, majority, etc.	I can communicate effectively with my tea, and when there is confusion or conflict, I can determine how best to improve my communication. I can communicate and collaborate effectively with my team, both online and in-person, during all stages of project development. I can identify, compare, and adapt a project management strategy to a specific task and group (e.g., waterfall, agile). I can be part of a team that delegates tasks with timelines and I can be accountable for my tasks. I can share an idea clearly and objectively with my team members. I can value the contributions of other team members. I can accept comments and criticism about my contributions and suggestions as our team develops solutions and work plans. I can identify key roles on effective teams, and I can evaluate the effectiveness of my role on my team. I can document and report on group decisions and rationale. I can document and report on tasks, timelines, progress, changes required, and project resolution.
Teacher Resources: https://nbed.sharepoint.com/sites/ComputerScience1102/	

SCO 1.4	**Students will apply the fundamentals of digital technology in relation to coding and computer science.**
Concepts and Content	**Achievement Indicators**
Fundamentals of Digital Technology in Relation to Coding: • to identify computer science and computer programming in everyday life and in relation to other disciplines • to identify current issues such as cybersecurity and the Internet • to identify and apply project management, software design and coding. • to identify key terms and aspects of computer science especially in relation to coding • to identify key components (e.g. hardware, operating system) needed for coding • to recognize the important role of debugging • to provide rationale and expectations regarding ethics and professional conduct. • to identify potential careers and career pathways using coding and computer science	I can identify uses of computer programming in my daily life. I can identify where computer science is used to complement other disciplines. I can identify areas where cybersecurity is important in computer science. I can identify current issues in computer science and coding. I can identify the Internet and some of its global impacts. I can identify and apply project management, software design, and coding. I can recognize where the terms coding, software development, and computer programming could be used, sometimes interchangeably. I can identify key elements of 'big data' and artificial intelligence. I can identify key elements of electronics, robotics, sensors, and the Internet of Things. I can identify key elements of graphics, animation, game design, and virtual and augmented reality. I recognize key differences in the hardware and uses of PCs, laptops, and mobile devices. I can identify key differences in operating systems (embedded, mobile, Windows, Linux, Mac). I can perform basic debugging and troubleshooting. I can create a successful coding project. I can provide rationale and expectations of the ethics and conduct of an IT professional. I can identify potential careers and career pathways that use coding and computer science, including post secondary opportunities.
Teacher Resources: https://nbed.sharepoint.com/sites/ComputerScience1102/	

GCO 2 Students will use computational thinking skills to analyze challenges and to create and evaluate solutions.

SCO 2.1	**Students will decompose a larger challenge into smaller manageable challenges.**
Concepts and Content	**Achievement indicators**
Decompose: • in the face of complex, ambiguous, non-trivial and difficult-to-grasp challenges • to analyze and determine why and how complex problems, challenges or tasks can be broken into smaller challenges that can each be solved in turn to be able to combine solutions to smaller challenges in order to solve the original larger challenge	I can work with complex, ambiguous, non-trivial, and difficult challenges. I can decompose a large complex challenge into smaller challenges that are more manageable and more easily solvable. I can explain why and how complex tasks can be broken into smaller ones that are simpler to solve I can combine the solutions to smaller challenges in a way that solves the original larger challenge. I can explain how decomposition is a useful technique in a variety of situations, beyond computers, networks, and coding.
Teacher Resources: **https://nbed.sharepoint.com/sites/ComputerScience1102/**	

GCO 2: Students will use computational thinking skills to analyze challenges and to create and evaluate solutions.

SCO 2.2	**Students will create repeatable solutions to manageable challenges.**
Concepts and Content	**Achievement Indicators**
Algorithmic Thinking: • to clearly define the steps, sequences and rules of a solution to a challenge • to assess a group of problems for similarities (elements in common) • to consider challenges where common elements are being solved over and over again, and therefore a common solution can be used Pattern Recognition and Automation: • to recognize situations and challenges where an automated solution can be applied • to recognize the sequences and rules involved in creating a repeatable solution (which may involve human and/or technology automation)	I can clearly define the steps to solve a challenge. I can create a solution based on the defined steps. I can assess groups of challenges for similarities, so that I can create common solutions (which may include loops, policies, or other repeatable solutions). I can secure a variety of digital devices using common patterns and solutions and I can document the steps in these solutions. I can guide a peer through the process of securing a digital device, including searching for vulnerabilities. I can adapt a solution to solve a challenge that has common elements, but is slightly different (and may involve human and/or technology automation). I can use and adapt an existing open source resources to solve a challenge.
Teacher Resources: https://nbed.sharepoint.com/sites/ComputerScience1102/	

GCO 2: Students will use computational thinking skills to analyze challenges and to create and evaluate solutions.

SCO 2.3	Students will represent, collect, and manage data to accomplish a task.
Concepts and Content	**Achievement Indicators**
Data Representation • to gather data related to a coding project • to use algorithms to generate and present data visualizations • to interpret data (using visualizations) to explain and/or prove a coding project will be successful	I can gather data related to a coding project and I can analyze that data to determine important elements. I can explain, using data, how a coding project will be successful.
Teacher Resources: https://nbed.sharepoint.com/sites/ComputerScience1102/	

SCO 2.4	Students will create and use algorithms to organize and analyze data and information
Concepts and Content	**Achievement Indicators**
Algorithmic Thinking • to evaluate the algorithms and program design of a coding project by analyzing data, and by identifying potential problems • to create a repeatable list of potential problems and solutions in a coding project • to explain how algorithms are used in coding projects, and the potential inaccuracies or security vulnerabilities Analysis • to troubleshoot a coding project, including debugging • to design a troubleshooting guide to help others use a coding project • to evaluate and improve a troubleshooting guide	I can use algorithms to help me analyze data from a coding project, including data visualizations. I can explain how algorithms might have inaccuracies and vulnerabilities when used by online systems. I can troubleshoot a coding project using a variety of tools. I can design a troubleshooting guide related to a coding project. I can evaluate and improve troubleshooting guides created others. I can sort data numerically and alphabetically. I can explain some differences between sorting algorithms such as bubble sort, selection sort and insertion sort.
Teacher Resources: https://nbed.sharepoint.com/sites/ComputerScience1102/	

GCO 2: Students will use computational thinking skills to analyze challenges and to create and evaluate solutions.

SCO 2.5	**Students will execute a solution and evaluate a solution's validity, efficiency, and effectiveness**
Concepts and Content	**Achievement Indicators**
Execute a Solution: • to complete the writing of a coding project that is a solution to a problem or challenge • to implement, execute and engage a coding project that is a solution to a problem or challenge Evaluate a Solution: • to evaluate, test or validate a coding project that is a solution to a problem or challenge • to assess or appraise a coding project for this solution's efficiency and effectiveness regarding the problem or challenge	I can complete a solution to a coding project challenge including the writing, debugging, and troubleshooting. I can implement and execute my solution to a coding project challenge. I can evaluate, test, or validate a coding project challenge's solution (that is either an existing resource or was made by me or my group). I can assess or appraise the efficiency and effectiveness of a solution to a coding project challenge's solution, whether it was created by me or not. I can validate that the algorithms and program design used to analyze data were appropriate, efficient, and effectively used in this solution.
Teacher Resources: https://nbed.sharepoint.com/sites/ComputerScience1102/	

GCO 3 Students will demonstrate skills and analyze computer science artifacts.

SCO 3.1	Students will analyze data needs and create artifacts that use data input, output and variables.

Concepts and Content	Achievement Indicators
Analyze a Project's Data Needs • to assess applicable data types and sizes • to assess applicable data methods of input and output • to assess applicable data manipulation and modification Create Coding Project Artifacts that Use Data • to create data variables • to receive input that is stored in these data variables • to manipulate and modify data stored in these variables to output applicable data stored in these variables	I can analyze and determine the necessary data based on a program design, including variable types and sizes as well as methods of input, manipulation, modification, and output. I can request and receive data as text and/or numeric input (through either the command-line or a form). I can save text input as a variable (string). I can save number input as a variable (integer or decimal). I can manipulate and/or modify text variables using string functions including the ability to replace and/or concatenate text and storing the result in the existing variable or in a new variable. I can manipulate and/or modify numeric variables by performing mathematical operations and storing the result in a new variable. I can send data as text and/or number output (to either the command-line or to a form).
Teacher Resources: **https://nbed.sharepoint.com/sites/ComputerScience1102/**	

SCO 3.2	**Students will analyze opportunities for modular code and create artifacts that use loops and other techniques that repeat and/or reuse code.**
Concepts and Content	**Achievement Indicators**
Analyze a Project's Opportunities for Reusing and/or Repeating Code • to assess applicable repetition methods such as loops • to assess applicable reuse methods such as functions, modules, classes and callbacks • to assess applicable reuse methods such as copying from previous projects and modifying for the current project Create Coding Project Artifacts that Reuse and/or Repeat Code • to create coding projects where code is used repeatedly through loops and functions • to create coding projects where code is reused by copying from previous projects and modifying for the current project	I can assess an applicable repetition method from possibilities such as FOR loops, WHILE loops, REPEAT loops. I can create a coding project artifact that uses my selected repetition method. I can assess an applicable reuse method from possibilities such as modifying code from a previous project, creating a function or class, or using a CALLBACK. I can create a coding project artifact that uses my selected reuse method.
Teacher Resources: https://nbed.sharepoint.com/sites/ComputerScience1102/	

SCO 3.3	**Students will analyze branching needs and create artifacts that use conditional statements.**
Concepts and Content	**Achievement Indicators**
Analyze a Project's Opportunities for Branching • to identify purposes and uses of various branching methods • to assess applicable branching methods such as conditional statements and conditional looping commands Create Coding Project Artifacts that Use Branching with Conditional Statements • to select and implement branching methods in code • to create coding projects where branching occurs using conditional statements	I can identify the purposes and uses of branching methods including conditional statements such as IF, ELSE, ELSEIF / ELIF, and SWITCH…CASE. I can assess an applicable branching method from possible conditional statements such as IF, ELSE, ELSEIF / ELIF, and SWITCH…CASE. I can create a coding project artifact that uses my selected branching method. I can identify the purposes and uses of branching methods, including conditional looping commands such as WHILE and REPEAT. I can assess an applicable repeating branching method from possible conditional looping commands such as WHILE and REPEAT. I can create a coding project artifact that uses my selected repeating branching method.
Teacher Resources: https://nbed.sharepoint.com/sites/ComputerScience1102/	
Note to teachers: *Microsoft products often refer to these as "decision structures" instead of "conditional statements".* *Microsoft products also use "SELECT ...CASE" where other languages use "SWITCH ...CASE".* *Python does not have a "SWITCH...CASE" and so designers must use ELIF or another solution.*	

SCO 3.4	**Students will analyze opportunities for abstraction and create artifacts that demonstrate abstraction in code.**
Concepts and Content	**Achievement Indicators**
Analyze a Project's Opportunities for Abstraction • to identify purposes and uses of abstraction • to assess applicable abstraction in data use such as determining which data can be safely ignored and which is relevant for use • to assess applicable abstraction in program design hiding unnecessary details from the user • to assess applicable abstraction in program design and coding where decomposition of challenges / tasks offers opportunities to repeat or reuse code Create Coding Project Artifacts that Use Abstraction • to select and implement abstraction in data design and data use • to select and implement abstraction in program design by hiding unnecessary details from the user • to select and implement abstraction in program design by decomposing large or complex tasks in to more manageable, repeatable and reusable code blocks	I can identify the purposes and uses of abstraction in data use. I can assess the potential for abstraction in data use in my project. I can select and implement coding that uses abstraction in data design and data use. I can identify the purposes and uses of abstraction in program design by hiding unnecessary details from the user. I can assess the potential for abstraction in my coding project's program design involving hiding unnecessary details from the user. I can select an abstraction technique and implement code in my project that uses abstraction by hiding unnecessary details from the user. I can identify the purposes and uses of abstraction in program design by decomposing complex / large tasks into smaller manageable ones. I can assess the potential for abstraction in program design where decomposing complex / large tasks into smaller manageable, repeatable and reusable tasks. I can select an abstraction technique and implement code in my project that uses abstraction to decompose complex / large tasks into smaller manageable, repeatable and reusable code blocks.
Teacher Resources: https://nbed.sharepoint.com/sites/ComputerScience1102/	

SCO 3.5	**Students will create and analyze artifacts with understandable code, helpful names, efficient comments and accompanying documentation.**
Concepts and Content	**Achievement Indicators**
Analyze a Project's Ability to be Understood • to identify purposes and importance of readable and understandable code, naming, comments and documentation. • to analyze and assess another project with regard to understandable code, naming, comments and documentation. • to analyze and assess one's own project with regard to understandable code, naming, comments and documentation. Create Coding Project Artifacts that can be Understood • to create a project that is consistent and highly understandable in its code, naming, comments and documentation	I can identify purposes and importance of readable and understandable code, naming, comments and documentation. I can use helpful descriptive names for variables, functions, modules, etc. I can use white-space effectively so that code is easier for people to read. I can create efficient yet descriptive comments to improve understanding. I can create an executive summary for a coding project. I can create a project summary including outline, flowchart and/or pseudocode, modules, and dependencies. I can create a project summary including a task list with assignments, especially if this was a group project. I can analyze and assess another project with regard to understandable code, naming, comments, and documentation. I can analyze and assess one's own project with regard to understandable code, naming, comments, and documentation. I can create coding project artifacts that are consistent and highly understandable in their code, naming, comments, and documentation
Teacher Resources: https://nbed.sharepoint.com/sites/ComputerScience1102/	

5. Bibliography

Common Content

Universal Design for Learning, Center for Applied Special Technology (CAST) http://www.cast.org/

Nelson, Louis Lord (2014). Design and Deliver: Planning and Teaching Using Universal Design for Learning. 1st Edition, Paul H. Brooks Publishing Co.

Subject Specific

Kaplan, F. M. (2017). *Dark territory: the secret history of cyber war*. New York: Simon & Schuster Paperbacks.

Stakhanova, N., & Stakhanov, O. (2017). *Have you been hacked yet?: how to protect your personal and financial information today*. Victoria, B.C.: Tellwell Talent.

6. Appendices

6.1 New Brunswick Global Competencies

Critical Thinking and Problem-Solving	Innovation, Creativity, and Entrepreneurship	Self-Awareness and Self-Management
• Engages in an inquiry process to solve problems • Acquires, processes, interprets, synthesizes, and critically analyzes information to make informed decisions (i.e., critical and digital literacy) • Selects strategies, resources, and tools to support their learning, thinking, and problem-solving • Evaluates the effectiveness of their choices • Sees patterns, makes connections, and transfers their learning from one situation to another, including real-world applications • Analyzes the functions and interconnections of social, ecological, and economic systems • Constructs, relates and applies knowledge to all domains of life, such as school, home, work, friends, and community • Solves meaningful, real-life, and complex problems by taking concrete steps to address issues and design and manage projects • Formulates and expresses questions to further their understanding, thinking, and problem-solving	• Displays curiosity, identifies opportunities for improvement and learning, and believes in their ability to improve • Views errors as part of the improvement process • Formulates and expresses insightful questions and opinions to generate novel ideas • Turns ideas into value for others by enhancing ideas or products to provide new-to-the-world or improved solutions to complex social, ecological, and economic problems or to meet a need in a community • Takes risks in their thinking and creating • Discovers through inquiry research, hypothesizing, and experimenting with new strategies or techniques • Seeks and makes use of feedback to clarify understanding, ideas, and products • Enhances concepts, ideas, or products through a creative process	• Has self-efficacy, sees themselves as learners, and believes that they can make life better for themselves and others • Develops a positive identity, sense of self, and purpose from their personal and cultural qualities • Develops and identifies personal, educational, and career goals, opportunities, and pathways • Monitors their progress • Perseveres to overcome challenges • Adapts to change and is resilient in adverse situations • Aware of, manages, and expresses their emotions, thoughts, and actions in order to understand themselves and others • Manages their holistic well-being (e.g., mental, physical, and spiritual) • Accurately self-assesses their current level of understanding or proficiency • Advocates for support based on their strengths, needs, and how they learn best • Manages their time, environment, and attention, including their focus, concentration, and engagement

Collaboration	Communication	Sustainability and Global Citizenship
• Participates in teams by establishing positive and respectful relationships, developing trust, and acting interdependently and with integrity • Learns from and contributes to the learning of others by co-constructing knowledge, meaning, and content • Assumes various roles on the team and respects a diversity of perspectives • Addresses disagreements and manages conflict in a sensitive and constructive manner • Networks with a variety of communities/groups • Appropriately uses an array of technology to work with others • Fosters social well-being, inclusivity, and belonging for themselves and others by creating and maintaining positive relationships with diverse groups of people • Demonstrates empathy for others in a variety of contexts	• Expresses themselves using the appropriate communication tools for the intended audience • Creates a positive digital identity • Communicates effectively in French and/or English and/or Mi'kmaq or Wolastoqey through a variety of media and in a variety of contexts • Gains knowledge about a variety of languages beyond their first and additional languages • Recognizes the strong connection between language and ways of knowing the world • Asks effective questions to create a shared communication culture, attend to understand all points of view, express their own opinions, and advocate for ideas	• Understands the interconnectedness of social, ecological, and economic forces, and how they affect individuals, societies, and countries • Recognizes discrimination and promotes principles of equity, human rights, and democratic participation • Understands Indigenous worldviews, traditions, values, customs, and knowledge • Learns from and with diverse people, develop cross-cultural understanding • Understands the forces that affect individuals and societies • Takes action and makes responsible decisions that support social settings, natural environments, and quality of life for all, now and in the future • Contributes to society and to the culture of local, national, global, and virtual communities in a responsible, inclusive, accountable, sustainable, and ethical manner • Participates in networks in a safe and socially responsible manner.
Foundation of Literacy and Numeracy		

6.2 Universal Design for Learning (UDL)

UDL helps meet the challenge of diversity by suggesting flexible instructional materials, techniques, and strategies that empower educators to meet these varied needs. UDL research demonstrates that the challenge of diversity can and must be met by making curriculum flexible and responsive to learner differences. UDL provides guidelines to minimize barriers and maximize learning for all.

Is there a form of **assistive technology** that could be used to enhance/facilitate this lesson?	Screen readers, screen magnifiers, speech-to-text, text-to-speech, etc.
Are there **materials which can appropriately challenge** readers to enhance this learning?	The online teacher resource web page contains a number of online sites, suggested services (free and commercial) and links to a NB teacher collaborative space.
Are there students in this group who cannot **access this learning (PLP background)** and whose needs I must revisit before teaching?	View previous PLP information for considerations
Are there other **choices** that can be provided in this learning opportunity?	Learning can be differentiated for outcomes as well as for depths of learning and methods of demonstrating learning
Is there another/a **variety of media** available? Only paper-based? Can it be listening? Can I add a visual component?	The online teacher resource provides a number of sites, including Khan Academy which focuses on video-based learning.
Can **movement** be involved?	Students can perform this learning on any device, although it is suggested to use a full monitor or even dual-monitor setup.

Grouping and regrouping?	Learning can be cooperative and in teams. Learning can be demonstrated using virtual machines and in games and competitions.
Teacher versus non- teacher centered? **Instructional design strategies** –…	Learning always revolves around the teacher, but opportunities exist for students to be more self-directed and self-paced using online resources and project-based learning. Students can self-initiate projects.
Opportunities for students to **propose variations** to the assignments/projects?	The initial tutorials are usually very straightforward and pre-set; however, once students demonstrate learning the fundamentals, then there are many opportunities for student project variation.
Use of **art /music / technology**?	Almost all student resources for this course are available online. There are many additional online resources, including web sites and YouTube videos.
Can I use **drama**? Art….	Role-playing and artistic expression can be used in many ways to explain or demonstrate learning about cybersecurity topics including ethical, psychological, sociological, and philosophical elements.
Is there a plan to support the student/s who might already know this subject matter? **Enrichment**	Students can prove prior learning and have opportunities to advance and enrich their own learning. This can be through self-paced tutorials or through self-initiated project proposals.
Does the **language level** need to be adjusted for the student to access this learning?	This course is very dependent on the use of the English language. While students can use online translators for context, the demonstrations of learning using coding are usually done in English. (Introductory lessons can use Scratch or other tools that are multi-lingual.)

Is there an **independent or collaborative activity-project** that would be better meet the needs of one or more students?	This course is largely based on tutorial work that leads to project-based learning. This work can be done independently or collaboratively, based on the needs of the student.
Are there any **experts** that I could bring into the classroom electronically or as a guest speaker?	There are many speakers available, locally and online, as well as documentary videos and local labour market data.
Have I linked the goal to as current event or a cultural event in the student's lives? Can I make the learning more **relevant**?	Computers, apps, and coding are topics that are relevant to every person, as electronics and computers are finding their way into everyday items like screw drivers, toasters, fridges, and clothing. This course starts slowly and builds quickly to cover the basics of computer programming, a.k.a. coding. Almost any activity or topic can have a coding application so there's no limits to the student's creativity.
Is there a **hands-on experience** that we could do to launch this lesson or this learning?	The learning is usually demonstrated through hands-on configuration of virtual machines in a safe online environment.

7. Resources

Learning Resources

All learning resources are being moved to: https://nbed.sharepoint.com/sites/ComputerScience1102/

Teacher Resources

Approaches to Teaching

It is recommended that teachers introduce computer science – including computer programming (aka coding) as well as basic computer operations and networks – through hands-on activities that are based on problems, challenges, and projects that become increasingly open-ended. These open-ended challenges avoid a single correct answer and instead have students weight the benefits, costs, risks, precedents, consequences, and side-effects of complex situations.

Activities may be based on challenges and problems that use games, electronics, robotics, Internet of Things (IoT), and other technologies that teachers determine to be useful as learning tools.

With the goal of fostering interest in computer science, teachers are encouraged to focus students on these hands-on activities, starting small and increasing in complexity. Rather than start from a theoretical computing science framework and have students memorize specific computing concepts, keywords, syntax or techniques, teachers are encouraged to provide challenges based on needs or problems which allow the learning of concepts, keywords, syntax and techniques along the way. This teaching strategy is recommended alongside other efforts to encourage a broader range of students to develop interests and skills in computer science. In addition, students report being more interested and engaged in projects that are self-created and impactful to those outside of the classroom.

Computer Science 110 teachers are encouraged to evolve from the lecture format to that of a guide, a coach and a mentor. The Computer Science 110 curriculum is designed with project-based learning in mind.

A fundamental principle of this course is that students assume responsibility for their own learning (ownership) through an inquiry-base/project-based learning approach. Since these strategies may be new to many students, teachers should discuss methods of organizing and brainstorming the big questions for inquiry, and introduce resources that help students critically address problems.

Students will know, and be able to use, strategies and processes to think creatively, understand deeply, conduct meaningful reflection, and solve problems independently and collaboratively. Students should be continuously aware of and planning for computer science projects while applying Global Competencies.

Being exposed to programs that involve collaboration and communication will develop important competencies mentioned above. Students should be encouraged to be resourceful and search the myriad of open source resources available on the internet to assist them in solving open-ended problems. Having students provide documentation within their computer science project work, as well as in the design, will help students to understand the meaning of functions, services, policies and processes involved, even if they are reusing resources (e.g., code snippets shared freely online).

Software Selection

Neither EECD nor this curriculum document require a specific computer programming (i.e. coding) language. We recommend that the language and related software will be what is best suited by balancing the following:

- the expertise of the teacher,
- the interests of the student,
- the opportunities within industry, and
- the language(s) that are suited to the project selected by the student (e.g. big data with R or Python).

It is hoped that teachers will have some experience in computer science and computer programming before teaching this course. Despite this, teachers should not feel the need to be experts in this field, as they encourage students to use resources and find mentors that allow their project work to exceed their expertise.

Regarding related software, one important consideration for teachers is the software used to create code, commonly referred to as an integrated development environment (IDE). While this is not specified or required, teachers should be aware that teachers in New Brunswick have consistently reported deeper programming skill is usually gained by students using a simplistic IDE, or even just a text editor (including those that colour code automatically).

Course Layout Option

Iterative approach to instruction:

As mentioned prior in this document; a feedback loop of instruction, hands-on learning and assessment are suggested for the application of skills into the larger scale project-based learning opportunities:

- Present one fundamental of computer coding (e.g. conditionals):
- What is the *computational thinking* behind this fundamental? (e.g. how “if” works)
- What *planning* surrounds this fundamental concept? (e.g. how “if” looks on a flowchart)
- What *syntax* does this fundamental require (e.g. “if (____) { }”)
- How is this *concept* used in my project? (e.g. where, why and how to include the “if” code)

Curriculum Flow (and Options)

The course could begin at sections *a, b* or *c*, as determined by the teacher based on the prior knowledge and experience of the students. Teachers can then assist students in choosing one or more items from section *d*. *Reminder: Please embed GCO's 1 and 2 into activities and projects designed to achieve the knowledge and skill outcomes of GCO 3.*

a. Visual Coding (Block Coding)

i. Conditionals (If, Else)
ii. Loops (For, Repeat Until, While Do)
iii. Variables (integer, decimal, strings)

b. Intro to Text-Based Coding

i. Output - Print "Hello World", GUI popup window
ii. Input - Receive and print a number or string
iii. Math - Add two numbers and create the sum (+, -, *, /, % ...)

c. Fundamentals of Text-Based Coding

i. Logic - Conditional Statements (i.e. Decision Structures)
ii. Loops - For, Repeat Until, While Do
iii. Classes / Procedures / Submodules (depending upon the language)
iv. Readable Code Conventions (naming, comments, whitespace, indentation)

d. Advanced Coding (Text-Based and/or Visual) *and/or Learning Another Language*

Topics (Optional / Just-in-Time):

Arrays Public/Private/Global Variables Hashes/Dictionaries File operations	Creating a game (i.e. Game Design) - Strategy, RPG, Action/Animation Creating a game using Artificial Intelligence (player vs computer)

Generating computer-based graphics (2D and/or 3D) Generating animations (for film/video) Use CodeAcademy or similar site to create an introductory coding tutorial	Learn web languages and frameworks (e.g., PHP in Drupal or Joomla, Ruby on Rails...) Learn HTML5 (HTML, CSS, and JavaScript) Connecting a Raspberry Pi to electronics using Scratch or Python

Sample Course Timetable

Timeline	Focus
Day 1	Focus on hands-on activities on day one: the more engaging, the better!
Weeks 1-3	Visual or text-based coding introducing syntax, logical flow and basic functions
End of Week 3	Students will have produced at least one coding artifact based on a project based approach.
Weeks 4-6	Introduction to basic built in functions in text-based coding
End of Week 6	Students will have produced a coding artifact based on a project based approach.
Weeks 7-9	Beginning of fundamentals of text-based coding.
End of Week 9	Students will have produced a coding artifact based on a project based approach.
Weeks 10-13	Students will complete fundamentals of text-based coding.
End of Week 13	Students will have produced a coding artifact based on a project-based approach.
Weeks 14-19	Mastery of fundamentals and/or extension into advanced coding topics.
End of Week 19	Students will produce a capstone project.