BIG IDEAS

Decomposition helps us solve difficult problems by managing complexity.	**Algorithms** are essential in solving problems computationally.	Programming is a tool that allows us to implement **computational thinking**.	**Solving problems** is a creative process.

Learning Standards

Curricular Competencies	Content
Students are expected to do the following: **Reasoning and modelling** • Develop **flexible thinking** to analyze and create algorithms • Explore, **analyze**, and apply mathematical ideas and computer science concepts using **reason**, **technology**, and **other tools** • **Model** with mathematics in **situational contexts** • **Think creatively** and with **curiosity and wonder** when exploring problems **Understanding and solving** • Develop, demonstrate, and apply conceptual understanding through experimentation, **inquiry**, and problem solving • **Visualize** to explore and illustrate computer science concepts and relationships • Apply **flexible and strategic approaches** to **solve problems** • Solve problems with **persistence and a positive disposition** • Engage in problem-solving experiences **connected** with place, story, cultural practices, and perspectives relevant to local First Peoples communities, the local community, and other cultures	*Students are expected to know the following:* • ways to represent **basic data types** • **basic programming concepts** • variable **scope** • ways to construct and evaluate **logical statements** • use of **control flow** to manipulate program execution • **development of algorithms** to solve problems in multiple ways • techniques for **operations** on and **searching** of arrays and lists • problem decomposition through **modularity** • uses of computing for **financial analysis** • ways to model **mathematical problems**

Learning Standards (continued)

Curricular Competencies	Content
Communicating and representing • **Explain and justify** mathematical ideas and **decisions** in **many ways** • **Represent** computer science ideas in concrete, pictorial, symbolic, and pseudocode forms • Use computer science and mathematical vocabulary and language to contribute to **discussions** in the classroom • Take risks when offering ideas in classroom **discourse** **Connecting and reflecting** • **Reflect** on mathematical and computational thinking • **Connect mathematical and computer science concepts** with each other, other areas, and personal interests • Use **mistakes** as **opportunities to advance learning** • **Incorporate** First Peoples worldviews, perspectives, **knowledge**, and **practices** to make connections with computer science concepts	

Big Ideas – Elaborations

- **Decomposition:**
  - dividing complex problems into parts that are easier to conceive, understand, and program

  Sample questions to support inquiry with students:
  - How do we break down a problem into several smaller, simpler pieces?
  - How do we know if a problem should be decomposed further?
  - Is there a better way to break a problem into smaller pieces and reuse code?
- **Algorithms:**
  - sets of rules or instructions that precisely define a sequence of operations

  Sample questions to support inquiry with students:
  - How does acting out a solution help us to develop an algorithm?
  - How is an algorithm formulated?
  - What makes one algorithm better than another algorithm?
  - How do we know that our algorithm is correct?
  - Can all problems be solved by a series of predefined steps?
- **computational thinking:**
  - a thought process that uses pattern recognition and decomposition to describe an algorithm in a way that a computer can execute

  Sample questions to support inquiry with students:
  - How do we decide which programming language to use in solving a specific problem?
  - Why is code readability important?
  - What factors affect code readability?
  - How much source code documentation is enough?
  - Are there patterns in the problem that can be generalized?
  - How do we recognize patterns that can be translated into rules?
- **Solving problems:**

  Sample questions to support inquiry with students:
  - How many different ways can this problem be solved?
  - How do we approach solving a problem in different ways?
  - Without knowing a solution, how do we start to solve a problem?

Curricular Competencies – Elaborations

- **flexible thinking:**
  - understanding that different algorithms can be used to solve the same problem
- **analyze:**
  - examine the structure of and connections between mathematical and computer science ideas (e.g., demonstrating the connection between theoretical and experimental probability through simulation)
- **reason:**
  - inductive and deductive reasoning
  - predictions, generalizations, conclusions drawn from experiences (e.g., with coding)
- **technology:**
  - graphing technology, dynamic geometry, calculators, virtual manipulatives, concept-based apps
  - can be used for a wide variety of purposes, including:
    - exploring and demonstrating mathematical relationships
    - organizing and displaying data
    - generating and testing inductive conjectures
    - mathematical modelling
- **other tools**
  - integrated development environments (IDE)
  - third-party libraries
  - visual code comparison tools to view code differences (e.g., Meld)
- **Model:**
  - use mathematical concepts and tools to solve problems and make decisions (e.g., in real-life and/or abstract scenarios)
  - take a complex, essentially non-mathematical scenario and figure out what mathematical concepts and tools are needed to make sense of it
- **situational contexts:**
  - including real-life scenarios and open-ended challenges that connect mathematics with everyday life
- **Think creatively:**
  - by being open to trying different strategies
  - refers to creative and innovative mathematical thinking rather than to representing math in a creative way, such as through art or music
- **curiosity and wonder:**
  - asking questions to further understanding or to open other avenues of investigation

Curricular Competencies – Elaborations

- **inquiry:**
  - includes structured, guided, and open inquiry
  - noticing and wondering
  - determining what is needed to make sense of and solve problems
- **Visualize:**
  - visualize data structures pictorially
  - use flow charts
  - use code visualization tools or websites (e.g., http://pythontutor.com/)
- **flexible and strategic approaches:**
  - using different algorithms to solve the same problem
  - designing algorithms that solve a class of problems rather than a single problem
  - deciding which programming patterns to use to solve a problem
  - choosing an effective strategy to solve a problem (e.g., guess and check, model, solve a simpler problem, use a chart, use diagrams, role-play)
- **solve problems:**
  - interpret a situation to identify a problem
  - apply mathematics to solve the problem
  - analyze and evaluate the solution in terms of the initial context
  - repeat this cycle until a solution makes sense
- **persistence and a positive disposition:**
  - not giving up when facing a challenge
  - problem solving with vigour and determination
- **connected:**
  - through daily activities, local and traditional practices, popular media and news events, cross-curricular integration
  - by posing and solving problems or asking questions about place, stories, and cultural practices
  - through cryptography (e.g., Navajo Code Talkers from WWII)
- **Explain and justify:**
  - use mathematical arguments to convince
  - includes anticipating consequences
- **decisions:**
  - Have students explore which of two scenarios they would choose and then defend their choice.

Curricular Competencies – Elaborations

- **many ways:**
  - including oral, written, pictures, use of technology
  - communicating effectively according to what is being communicated and to whom
- **Represent:**
  - using models, tables, flow charts, words, numbers, symbols
  - connecting meanings among various representations
  - using concrete materials and dynamic interactive technology
- **discussions:**
  - partner talks, small-group discussions, teacher-student conferences
- **discourse:**
  - is valuable for deepening understanding of concepts
  - can help clarify students' thinking, even if they are not sure about an idea or have misconceptions
- **Reflect:**
  - share the mathematical and computational thinking of self and others, including evaluating strategies and solutions, extending, posing new problems and questions
- **Connect mathematical and computer science concepts:**
  - to develop a sense of how computer science helps us understand the world around us (e.g., daily activities, local and traditional practices, popular media and news events, social justice, cross-curricular integration)
- **mistakes:**
  - include syntax, semantic, run-time, and logic errors
- **opportunities to advance learning:**
  - by:
    - analyzing errors to discover misunderstandings
    - making adjustments in further attempts (e.g., debugging)
    - identifying not only mistakes but also parts of a solution that are correct
- **Incorporate:**
  - by:
    - collaborating with Elders and knowledge keepers among local First Peoples
    - exploring the First Peoples Principles of Learning (e.g., Learning is holistic, reflexive, reflective, experiential, and relational [focused on connectedness, on reciprocal relationships, and a sense of place]; Learning involves patience and time)
    - making explicit connections with learning mathematics
    - exploring cultural practices and knowledge of local First Peoples and identifying mathematical connections

Curricular Competencies – Elaborations

- **knowledge:**
  - local knowledge and cultural practices that are appropriate to share and that are non-appropriated
- **practices:**
  - Bishop's cultural practices: counting, measuring, locating, designing, playing, explaining
  - Aboriginal Education Resources
  - *Teaching Mathematics in a First Nations Context*, FNESC

Content – Elaborations

- **basic data types:**
  - number systems (e.g., binary, hexadecimal)
  - strings, integers, characters, floating point
- **basic programming concepts:**
  - variables, constants, mathematical operations, input/output, generating random numbers
- **scope:**
  - local versus global
- **logical statements:**
  - logical operators (AND, OR, NOT)
  - relational operators (<, >, <=, >=, ==, !=, or <>)
  - logical equivalences (e.g., De Morgan's laws), simplification of logical statements, truth tables
- **control flow:**
  - decision structures (e.g., if-then-else)
  - loops (e.g., for, while, nested loops)
- **development of algorithms:**
  - step-wise refinement, pseudocode or flowcharts, translating between pseudocode and code and vice versa

Content – Elaborations

- **operations:**
  - append, remove, insert, delete
- **searching:**
  - searching algorithms (e.g., linear and binary searches)
- **modularity:**
  - use of methods/functions to reduce complexity, reuse code, and use function parameters
  - return values
- **financial analysis:**
  - time value of money, appreciation/depreciation, mortgage amortization
  - modify the variables of a financial scenario to run a "what-if" analysis on them (e.g., compare different monthly payments, term lengths, interest rates)
- **mathematical problems:**
  - estimate theoretical probability through simulation
  - represent finite sequences and series
  - solve a system of linear equations, exponential growth/decay
  - solve a polynomial equation
  - calculate statistical values such as frequency, central tendencies, standard deviation of large data set
  - compute greatest common factor/least common multiples

