{MAT 335} AP Computer Science

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Date** | **hours** | **Units** | **Vocabulary** | **Standards Resources** | |  |  |  |
| September  September | Lab 1: Click Alonzo Game  (40–80 minutes) for required pages  Lab 2: Gossip  (90–180 minutes)  Lab 3: Modern Art with Polygons  (120–240 minutes)  Lab 4: Protecting Your Privacy  (120–240 minutes)  Lab 5: Follow the Leader  (35–70 minutes)  Optional Projects | [Unit 1: Introduction to Programming](https://bjc.edc.org/bjc-r/topic/topic.html?topic=nyc_bjc/1-intro-loops.topic&course=bjc4nyc.html&novideo&noassignment) Lab 1: Click Alonzo Game [Getting Started with Snap!](https://bjc.edc.org/bjc-r/cur/programming/1-introduction/1-building-an-app/1-creating-a-snap-account.html?topic=nyc_bjc%2F1-intro-loops.topic&course=bjc4nyc.html&novideo&noassignment)  [Programming a Game](https://bjc.edc.org/bjc-r/cur/programming/1-introduction/1-building-an-app/2-start-your-first-snap-app.html?topic=nyc_bjc%2F1-intro-loops.topic&course=bjc4nyc.html&novideo&noassignment)  [Sharing Your Game](https://bjc.edc.org/bjc-r/cur/programming/1-introduction/1-building-an-app/3-loading-mobile-device.html?topic=nyc_bjc%2F1-intro-loops.topic&course=bjc4nyc.html&novideo&noassignment)  [Giving the Player Feedback](https://bjc.edc.org/bjc-r/cur/programming/1-introduction/1-building-an-app/4-keeping-score.html?topic=nyc_bjc%2F1-intro-loops.topic&course=bjc4nyc.html&novideo&noassignment)  [Improving Your Game](https://bjc.edc.org/bjc-r/cur/programming/1-introduction/1-building-an-app/5-finish-your-first-snap-app.html?topic=nyc_bjc%2F1-intro-loops.topic&course=bjc4nyc.html&novideo&noassignment) Lab 2: Gossip [Pair Programming](https://bjc.edc.org/bjc-r/cur/programming/1-introduction/2-gossip-and-greet/1-pair-programming.html?topic=nyc_bjc%2F1-intro-loops.topic&course=bjc4nyc.html&novideo&noassignment)  [Making Programs Talk](https://bjc.edc.org/bjc-r/cur/programming/1-introduction/2-gossip-and-greet/2-gossip.html?topic=nyc_bjc%2F1-intro-loops.topic&course=bjc4nyc.html&novideo&noassignment)  [Customizing and Debugging](https://bjc.edc.org/bjc-r/cur/programming/1-introduction/2-gossip-and-greet/3-customizing.html?topic=nyc_bjc%2F1-intro-loops.topic&course=bjc4nyc.html&novideo&noassignment)  [Making Your Own Block](https://bjc.edc.org/bjc-r/cur/programming/1-introduction/2-gossip-and-greet/4-making-a-new-block.html?topic=nyc_bjc%2F1-intro-loops.topic&course=bjc4nyc.html&novideo&noassignment)  [Adding Variety to **Gossip**](https://bjc.edc.org/bjc-r/cur/programming/1-introduction/2-gossip-and-greet/5-if-else.html?topic=nyc_bjc%2F1-intro-loops.topic&course=bjc4nyc.html&novideo&noassignment) Lab 3: Modern Art with Polygons [Exploring Motion](https://bjc.edc.org/bjc-r/cur/programming/1-introduction/3-drawing/1-exploring-motion.html?topic=nyc_bjc%2F1-intro-loops.topic&course=bjc4nyc.html&novideo&noassignment)  [Angles and Turning](https://bjc.edc.org/bjc-r/cur/programming/1-introduction/3-drawing/2-exploring-snap-drawing-motion.html?topic=nyc_bjc%2F1-intro-loops.topic&course=bjc4nyc.html&novideo&noassignment)  [Blocks with Inputs](https://bjc.edc.org/bjc-r/cur/programming/1-introduction/3-drawing/3-blocks-with-inputs.html?topic=nyc_bjc%2F1-intro-loops.topic&course=bjc4nyc.html&novideo&noassignment)  [Modify Your Pinwheel](https://bjc.edc.org/bjc-r/cur/programming/1-introduction/3-drawing/4-modify-your-pinwheel.html?topic=nyc_bjc%2F1-intro-loops.topic&course=bjc4nyc.html&novideo&noassignment)  [Using **Pinwheel** to Make **Polygon**](https://bjc.edc.org/bjc-r/cur/programming/1-introduction/3-drawing/5-remix-your-pinwheel.html?topic=nyc_bjc%2F1-intro-loops.topic&course=bjc4nyc.html&novideo&noassignment)  [Looping with a Counter](https://bjc.edc.org/bjc-r/cur/programming/1-introduction/3-drawing/6-the-for-block.html?topic=nyc_bjc%2F1-intro-loops.topic&course=bjc4nyc.html&novideo&noassignment)  [Keeping a Programming Journal](https://bjc.edc.org/bjc-r/cur/programming/1-introduction/3-drawing/7-programming-journal.html?topic=nyc_bjc%2F1-intro-loops.topic&course=bjc4nyc.html&novideo&noassignment) Lab 4: Protecting Your Privacy[Your Image in the Cloud](https://bjc.edc.org/bjc-r/cur/programming/1-introduction/4-privacy/1-your-image-in-the-cloud.html?topic=nyc_bjc%2F1-intro-loops.topic&course=bjc4nyc.html&novideo&noassignment) [Examining Privacy](https://bjc.edc.org/bjc-r/cur/programming/1-introduction/4-privacy/2-examining-privacy.html?topic=nyc_bjc%2F1-intro-loops.topic&course=bjc4nyc.html&novideo&noassignment)  [Tanya Rider's Story](https://bjc.edc.org/bjc-r/cur/programming/1-introduction/4-privacy/3-tanya-rider.html?topic=nyc_bjc%2F1-intro-loops.topic&course=bjc4nyc.html&novideo&noassignment)  [Innovations and Privacy](https://bjc.edc.org/bjc-r/cur/programming/1-introduction/4-privacy/4-privacy-affected.html?topic=nyc_bjc%2F1-intro-loops.topic&course=bjc4nyc.html&novideo&noassignment) Lab 5: Follow the Leader [Sprite Following the Mouse](https://bjc.edc.org/bjc-r/cur/programming/1-introduction/5-follow-the-leader/1-following-sprites.html?topic=nyc_bjc%2F1-intro-loops.topic&course=bjc4nyc.html&novideo&noassignment)  [Sprite Following a Sprite](https://bjc.edc.org/bjc-r/cur/programming/1-introduction/5-follow-the-leader/2-sprite-interaction.html?topic=nyc_bjc%2F1-intro-loops.topic&course=bjc4nyc.html&novideo&noassignment)  \*if time there are optional projects to be completed | Sprites – costumes – transparency  Lists – strings – concatenation- substring – debugging – procedures – reporters – commands – expressions - values  Algorithm – pseudocode – parameter – argument – iteration  Personal Identifiable Information (PII)  Infinite loop – code segment | **CRD-2.B.1:** A *program* is a collection of program statements that performs a specific task when run by a computer. A program is often referred to as *software*.  **CRD-2.C.2:** An *event* is associated with an action and supplies input data to a program.  **CRD-2.C.3:** Events can be generated when a key is pressed, a mouse is clicked, a program is started, or by any other defined action that affects the flow of execution.  **CRD-2.C.5:** In event-driven programming, program statements are executed when triggered rather than through the sequential flow of control. | [Introduction to Teacher Guide (edc.org)](https://bjc.edc.org/bjc-r/cur/teaching-guide/U1/overview.html?course=bjc4nyc_teacher.html&novideo&noassignment)  [Teaching Guide for Unit 1: Introduction to Programming (edc.org)](https://bjc.edc.org/bjc-r/topic/topic.html?topic=nyc_bjc/1-teaching-guide.topic&course=bjc4nyc_teacher.html&novideo&noassignment) | |  |  |
|  | Lab 1: Games  (75–150 minutes)  Lab 2: Making Lists  (75–150 minutes)  Lab 3: Making Decisions  (140–280 minutes)  Lab 4: Making Computers Do Math  (60–120 minutes)  Lab 5: Copyright  (50–100 minutes)  Optional Projects | [Unit 2: Abstraction (edc.org)](https://bjc.edc.org/bjc-r/topic/topic.html?topic=nyc_bjc/2-conditionals-abstraction.topic&course=bjc4nyc.html&novideo&noassignment)  **Lab 1: Games**  [Starting a Number Guessing Game](https://bjc.edc.org/bjc-r/cur/programming/2-complexity/1-variables-games/1-number-guessing-game.html?topic=nyc_bjc%2F2-conditionals-abstraction.topic&course=bjc4nyc.html&novideo&noassignment)  [Checking the Player's Guess](https://bjc.edc.org/bjc-r/cur/programming/2-complexity/1-variables-games/2-checking-player-guess.html?topic=nyc_bjc%2F2-conditionals-abstraction.topic&course=bjc4nyc.html&novideo&noassignment)  [Debugging and Extending Your Number Guessing Game](https://bjc.edc.org/bjc-r/cur/programming/2-complexity/1-variables-games/3-debugging-extending.html?topic=nyc_bjc%2F2-conditionals-abstraction.topic&course=bjc4nyc.html&novideo&noassignment)  [Keeping Score with Global Variables](https://bjc.edc.org/bjc-r/cur/programming/2-complexity/1-variables-games/4-keeping-score.html?topic=nyc_bjc%2F2-conditionals-abstraction.topic&course=bjc4nyc.html&novideo&noassignment)  [Choosing a Costume](https://bjc.edc.org/bjc-r/cur/programming/2-complexity/1-variables-games/5-choosing-avatar.html?topic=nyc_bjc%2F2-conditionals-abstraction.topic&course=bjc4nyc.html&novideo&noassignment) Lab 2: Making Lists [Shopping List App](https://bjc.edc.org/bjc-r/cur/programming/2-complexity/2-data-structures-art/1-shopping-list-app.html?topic=nyc_bjc%2F2-conditionals-abstraction.topic&course=bjc4nyc.html&novideo&noassignment)  [Planning a Quiz App](https://bjc.edc.org/bjc-r/cur/programming/2-complexity/2-data-structures-art/2-quizzes.html?topic=nyc_bjc%2F2-conditionals-abstraction.topic&course=bjc4nyc.html&novideo&noassignment)  [Checking Each Quiz Answer](https://bjc.edc.org/bjc-r/cur/programming/2-complexity/2-data-structures-art/3-traversing-list.html?topic=nyc_bjc%2F2-conditionals-abstraction.topic&course=bjc4nyc.html&novideo&noassignment) Lab 3: Making Decisions [What's a Predicate?](https://bjc.edc.org/bjc-r/cur/programming/2-complexity/3-predicates/1-what-is-predicate.html?topic=nyc_bjc%2F2-conditionals-abstraction.topic&course=bjc4nyc.html&novideo&noassignment)  [Combining Predicates](https://bjc.edc.org/bjc-r/cur/programming/2-complexity/3-predicates/2-combining-predicates.html?topic=nyc_bjc%2F2-conditionals-abstraction.topic&course=bjc4nyc.html&novideo&noassignment)  [Combining Conditionals](https://bjc.edc.org/bjc-r/cur/programming/2-complexity/3-predicates/3-combining-conditionals.html?topic=nyc_bjc%2F2-conditionals-abstraction.topic&course=bjc4nyc.html&novideo&noassignment)  [Boolean Expression Experiments](https://bjc.edc.org/bjc-r/cur/programming/2-complexity/3-predicates/4-boolean-experiments.html?topic=nyc_bjc%2F2-conditionals-abstraction.topic&course=bjc4nyc.html&novideo&noassignment)  [**Keep**ing Items from a List](https://bjc.edc.org/bjc-r/cur/programming/2-complexity/3-predicates/5-keeping-list-items.html?topic=nyc_bjc%2F2-conditionals-abstraction.topic&course=bjc4nyc.html&novideo&noassignment)  [Solving a Word Puzzle](https://bjc.edc.org/bjc-r/cur/programming/2-complexity/3-predicates/6-solving-word-puzzle.html?topic=nyc_bjc%2F2-conditionals-abstraction.topic&course=bjc4nyc.html&novideo&noassignment) Lab 4: Making Computers Do Math [The **Mod** Operator](https://bjc.edc.org/bjc-r/cur/programming/2-complexity/4-making-computers-do-math/1-mod-operator.html?topic=nyc_bjc%2F2-conditionals-abstraction.topic&course=bjc4nyc.html&novideo&noassignment)  [Making a Mathematical Library](https://bjc.edc.org/bjc-r/cur/programming/2-complexity/4-making-computers-do-math/2-math-predicates.html?topic=nyc_bjc%2F2-conditionals-abstraction.topic&course=bjc4nyc.html&novideo&noassignment)  [More Mathematical Reporters](https://bjc.edc.org/bjc-r/cur/programming/2-complexity/4-making-computers-do-math/3-other-math-reporters.html?topic=nyc_bjc%2F2-conditionals-abstraction.topic&course=bjc4nyc.html&novideo&noassignment) Lab 5: Copyrights [Copyright](https://bjc.edc.org/bjc-r/cur/programming/2-complexity/5-copyrights/1-copyright.html?topic=nyc_bjc%2F2-conditionals-abstraction.topic&course=bjc4nyc.html&novideo&noassignment)  [Fair Use](https://bjc.edc.org/bjc-r/cur/programming/2-complexity/5-copyrights/2-fairuse.html?topic=nyc_bjc%2F2-conditionals-abstraction.topic&course=bjc4nyc.html&novideo&noassignment)  [The Social Bargain](https://bjc.edc.org/bjc-r/cur/programming/2-complexity/5-copyrights/3-bargain.html?topic=nyc_bjc%2F2-conditionals-abstraction.topic&course=bjc4nyc.html&novideo&noassignment)  [Copyrights and Computers](https://bjc.edc.org/bjc-r/cur/programming/2-complexity/5-copyrights/4-digital.html?topic=nyc_bjc%2F2-conditionals-abstraction.topic&course=bjc4nyc.html&novideo&noassignment)  [Software as Copyright Enforcement](https://bjc.edc.org/bjc-r/cur/programming/2-complexity/5-copyrights/5-drm.html?topic=nyc_bjc%2F2-conditionals-abstraction.topic&course=bjc4nyc.html&novideo&noassignment) | |  |  |  | | --- | --- | --- | | Variable – local variable – predicate – Boolean – conditional – global variable – initialize – list- index  Element – sublist – data type – primitive data type – constructor – selector – data abstracting – composition of functions – traversing – abstract data type |  |  | | Domain – range – input – output – sequencing – selection – iteration – string- index – nestled conditional statement  Software library – application program interface (API)  Creative commons – free software – open source – open access |  |  | |  |  |  | | **AAP-2.C:** Evaluate expressions that use arithmetic operators.  **AAP-2.E:** For relationships between two variables, expressions, or values:  Write expressions using relational operators.  Evaluate expressions that use relational operators.  **AAP-2.F:** For relationships between Boolean values:  Write expressions using logical operators.  Evaluate expressions that use logic operators.  **AAP-2.H:** For selection:  Write conditional statements.  Determine the result of conditional statements.  **AAP-2.I:** For nested selection:  Write nested conditional statements.  Determine the result of nested conditional statements.  **AAP-2.L:** Compare multiple algorithms to determine if they yield the same side effect or result.  **AAP-1.C.3:** An index is a common method for referencing the elements in a list or string using natural numbers.  **AAP-1.C.4:** A string is an ordered sequence of characters.  **AAP-2.A.4:** Every algorithm can be constructed using combinations of sequencing, selection, and iteration.  **AAP-2.B.1:** Sequencing is the application of each step of an algorithm in the order in which the code statements are given.  **AAP-2.G.1:** Selection determines which parts of an algorithm are executed based on a condition being true or false  **AAP-2.H.1:** Conditional statements or “if-statements” affect the sequential flow of control by executing different statements based on the value of a Boolean expression.  **AAP-2.I.1:** Nested conditional statements consist of conditional statements within conditional statements.  **AAP-2.L.1:** Algorithms can be written in different ways and still accomplish the same tasks.  **AAP-2.L.3:** Some conditional statements can be written as equivalent Boolean expressions.  **AAP-2.L.4:** Some Boolean expressions can be written as equivalent conditional statements.  **AAP-2.L.5:** Different algorithms can be developed or used to solve the same problem. | [Teaching Guide for Unit 2: Abstraction (edc.org)](https://bjc.edc.org/bjc-r/topic/topic.html?topic=nyc_bjc/2-teaching-guide.topic&course=bjc4nyc_teacher.html&novideo&noassignment) | |  |  |
| October  October/ November | Lab 1: Dealing with Complexity  (110–220 minutes)  Lab 2: Contact List  (85–170 minutes) for required pages | [Unit 3: Data Structures (edc.org)](https://bjc.edc.org/bjc-r/topic/topic.html?topic=nyc_bjc/3-lists.topic&course=bjc4nyc.html&novideo&noassignment) Lab 1: Dealing with Complexity [Robot in a Maze](https://bjc.edc.org/bjc-r/cur/programming/3-lists/1-abstraction/1-robot-in-a-maze.html?topic=nyc_bjc%2F3-lists.topic&course=bjc4nyc.html&novideo&noassignment)  [Fractal Art](https://bjc.edc.org/bjc-r/cur/programming/3-lists/1-abstraction/2-fractal-art.html?topic=nyc_bjc%2F3-lists.topic&course=bjc4nyc.html&novideo&noassignment)  [Using Abstraction to Nest Triangles](https://bjc.edc.org/bjc-r/cur/programming/3-lists/1-abstraction/3-fractal-art-recursive.html?topic=nyc_bjc%2F3-lists.topic&course=bjc4nyc.html&novideo&noassignment)  [Brick Wall](https://bjc.edc.org/bjc-r/cur/programming/3-lists/1-abstraction/4-brick-wall.html?topic=nyc_bjc%2F3-lists.topic&course=bjc4nyc.html&novideo&noassignment)  [Building a Tic-Tac-Toe Board](https://bjc.edc.org/bjc-r/cur/programming/3-lists/1-abstraction/5-building-tic-tac-toe.html?topic=nyc_bjc%2F3-lists.topic&course=bjc4nyc.html&novideo&noassignment)  [Debugging Recap](https://bjc.edc.org/bjc-r/cur/programming/3-lists/1-abstraction/6-debugging-recap.html?topic=nyc_bjc%2F3-lists.topic&course=bjc4nyc.html&novideo&noassignment) Lab 2: Contact List[Creating the Contact ADT](https://bjc.edc.org/bjc-r/cur/programming/3-lists/2-contact-list/1-build-the-list.html?topic=nyc_bjc%2F3-lists.topic&course=bjc4nyc.html&novideo&noassignment) [Adding Contact Data](https://bjc.edc.org/bjc-r/cur/programming/3-lists/2-contact-list/2-adding-contact.html?topic=nyc_bjc%2F3-lists.topic&course=bjc4nyc.html&novideo&noassignment)  [Adding Birthdays](https://bjc.edc.org/bjc-r/cur/programming/3-lists/2-contact-list/3-modifying-adt.html?topic=nyc_bjc%2F3-lists.topic&course=bjc4nyc.html&novideo&noassignment)  [Selecting Specific Data](https://bjc.edc.org/bjc-r/cur/programming/3-lists/2-contact-list/4-looking-up-data.html?topic=nyc_bjc%2F3-lists.topic&course=bjc4nyc.html&novideo&noassignment)  [Transforming Every List Item](https://bjc.edc.org/bjc-r/cur/programming/3-lists/2-contact-list/5-mapping-over-list.html?topic=nyc_bjc%2F3-lists.topic&course=bjc4nyc.html&novideo&noassignment) | Recursion – abstraction – procedural abstraction – draw brick – modularity – clone – debugging  Input – output – modularity – high order function | **AAP-2.A:** Express an algorithm that uses sequencing without using a programming language.  **AAP-2.G:** Express an algorithm that uses selection without using a programming language.  **AAP-2.J:** Express an algorithm that uses iteration without using a programming language.  **AAP-3.B:** Explain how the use of procedural abstraction manages complexity in a program.  **AAP-3.C:** Develop procedural abstractions to manage complexity in a program by writing procedures.  **AAP-1.A.2:** Using meaningful variable names helps with the readability of program code and understanding of what values are represented by the variables.  **AAP-2.B.7:** Clarity and readability are important considerations when expressing an algorithm in a programming language.  **AAP-2.M.2:** Knowledge of existing algorithms can help in constructing new ones. Some existing algorithms include: determining a robot's path through a maze  **AAP-3.B.1:** One common type of abstraction is procedural abstraction, which provides a name for a process and allows a procedure to be used only knowing what it does, not how it does it.  **AAP-3.B.5:** Using parameters allows procedures to be generalized, enabling the procedures to be reused with a range of input values or arguments.  **CRD-2.C:** Identify input(s) to a program.  **CRD-2.D:** Identify output(s) produced by a program.  **AAP-1.D:** For data abstraction:  Explain how the use of data abstraction manages complexity in program code.  **CRD-2.C.1:** *Program input* is data sent to a computer for processing by a program. Input can come in a variety of forms, such as tactile, audio, visual, or text.  **CRD-2.C.4:** Inputs usually affect the output produced by a program.  **CRD-2.C.6:** Input can come from a user or other programs.  **CRD-2.D.1:** *Program output* is any data sent from a program to a device. Program output can come in a variety of forms, such as tactile, audio, visual, or text.  **CRD-2.D.2:** Program output is usually based on a program’s input or prior state (e.g., internal values).  **DAT-2.D.3:** Search tools are useful for efficiently finding information.  **AAP-1.D.5:** Data abstractions often contain different types of elements.  **AAP-3.B.2:** Procedural abstraction allows a solution to a large problem to be based on the solution of smaller subproblems. This is accomplished by creating procedures to solve each of the subproblems.  **AAP-3.B.3:** The subdivision of a computer program into separate subprograms is called *modularity*. | [Teaching Guide for Unit 3: Data Structures (edc.org)](https://bjc.edc.org/bjc-r/topic/topic.html?topic=nyc_bjc/3-teaching-guide.topic&course=bjc4nyc_teacher.html&novideo&noassignment) | |  |  |
| November | Lab 3: Tic-Tac-Toe  no required pages  Lab 4: Robots and Artificial Intelligence  (85–170 minutes) for required pages  Lab 5: Computers and Work  (70–140 minutes)  Optional Projects | **Lab 3: Tic-Tac-Toe**  [Remembering the Moves and Possible Wins](https://bjc.edc.org/bjc-r/cur/programming/3-lists/3-tic-tac-toe/1-find-ties.html?topic=nyc_bjc%2F3-lists.topic&course=bjc4nyc.html&novideo&noassignment)  [Detecting A Win](https://bjc.edc.org/bjc-r/cur/programming/3-lists/3-tic-tac-toe/2-checking-for-a-win.html?topic=nyc_bjc%2F3-lists.topic&course=bjc4nyc.html&novideo&noassignment)  [Checking for Ties](https://bjc.edc.org/bjc-r/cur/programming/3-lists/3-tic-tac-toe/3-checking-for-ties.html?topic=nyc_bjc%2F3-lists.topic&course=bjc4nyc.html&novideo&noassignment)  **Lab 4: Robots and Artificial Intelligence**  [What is AI?](https://bjc.edc.org/bjc-r/cur/programming/3-lists/4-robots-ai/1-what-is-ai.html?topic=nyc_bjc%2F3-lists.topic&course=bjc4nyc.html&novideo&noassignment)  [Robots and Humans](https://bjc.edc.org/bjc-r/cur/programming/3-lists/4-robots-ai/2-robots-and-humans.html?topic=nyc_bjc%2F3-lists.topic&course=bjc4nyc.html&novideo&noassignment)  [Implications of AI](https://bjc.edc.org/bjc-r/cur/programming/3-lists/4-robots-ai/3-implications.html?topic=nyc_bjc%2F3-lists.topic&course=bjc4nyc.html&novideo&noassignment)  [Recent Breakthroughs](https://bjc.edc.org/bjc-r/cur/programming/3-lists/4-robots-ai/4-breakthroughs-possibilities.html?topic=nyc_bjc%2F3-lists.topic&course=bjc4nyc.html&novideo&noassignment)  **Lab 5: Computers and Work**  [Past and Future](https://bjc.edc.org/bjc-r/cur/programming/3-lists/5-work/1-past-and-future.html?topic=nyc_bjc%2F3-lists.topic&course=bjc4nyc.html&novideo&noassignment)  [Working Conditions](https://bjc.edc.org/bjc-r/cur/programming/3-lists/5-work/2-working-conditions.html?topic=nyc_bjc%2F3-lists.topic&course=bjc4nyc.html&novideo&noassignment)  [Working Remotely](https://bjc.edc.org/bjc-r/cur/programming/3-lists/5-work/3-working-remotely.html?topic=nyc_bjc%2F3-lists.topic&course=bjc4nyc.html&novideo&noassignment) | Sprite variable  Artificial intelligence (AI) | **IOC-1.A:** Explain how an effect of a computing innovation can be both beneficial and harmful.  **IOC-1.D:** Explain how bias exists in computing innovations.  **IOC-1.B.1:** Computing innovations can be used in ways that their creators had not originally intended: Machine learning and data mining have enabled innovation in medicine, business, and science, but information discovered in this way has also been used to discriminate against groups of individuals.  **IOC-1.D.1:** Computing innovations can reflect existing human biases because of biases written into the algorithms or biases in the data used by the innovation.  **IOC-1.D.2:** Programmers should take action to reduce bias in algorithms used for computing innovations as a way of combating existing human biases.  **IOC-1.D.3:** Biases can be embedded at all levels of software development.  **IOC-1.F.11:** Computing innovations can raise legal and ethical concerns. Some examples of these include:  the development of algorithms that include bias  **IOC-1.A:** Explain how an effect of a computing innovation can be both beneficial and harmful.  **IOC-1.A.2:** The way people complete tasks often changes to incorporate new computing innovations. | [Teaching Guide for Unit 3: Data Structures (edc.org)](https://bjc.edc.org/bjc-r/topic/topic.html?topic=nyc_bjc/3-teaching-guide.topic&course=bjc4nyc_teacher.html&novideo&noassignment) | |  |  |
| November 11th |  | **10 WEEK MARKING PERIOD CLOSES** | **10 WEEK MARKING PERIOD CLOSES** | **10 WEEK MARKING PERIOD CLOSES** | **10 WEEK MARKING PERIOD CLOSES** | |  |  |
| November  **20 week making period** | AP CSP Create Task  (at least 12 hours required) | **AP CSP Create Task**  [Using a Development Process to Organize Your Coding](https://bjc.edc.org/bjc-r/cur/performance-tasks/create-task/1-development-process.html?topic=nyc_bjc%2Fcreate-task.topic&course=bjc4nyc.html&novideo&noassignment)  [Choosing Your Project](https://bjc.edc.org/bjc-r/cur/performance-tasks/create-task/2-choosing-your-project.html?topic=nyc_bjc%2Fcreate-task.topic&course=bjc4nyc.html&novideo&noassignment)  [Implementing Your Development Process](https://bjc.edc.org/bjc-r/cur/performance-tasks/create-task/3-implementing.html?topic=nyc_bjc%2Fcreate-task.topic&course=bjc4nyc.html&novideo&noassignment)  [Testing Your Project](https://bjc.edc.org/bjc-r/cur/performance-tasks/create-task/4-testing.html?topic=nyc_bjc%2Fcreate-task.topic&course=bjc4nyc.html&novideo&noassignment)  [Communicating About Your Project](https://bjc.edc.org/bjc-r/cur/performance-tasks/create-task/5-communicating.html?topic=nyc_bjc%2Fcreate-task.topic&course=bjc4nyc.html&novideo&noassignment)  [Evaluating Your Work](https://bjc.edc.org/bjc-r/cur/performance-tasks/create-task/6-evaluating.html?topic=nyc_bjc%2Fcreate-task.topic&course=bjc4nyc.html&novideo&noassignment) | ALL PREVIOUS VOCABULAY | **CRD-2.E:** Develop a program using a development process.  **CRD-2.F:** Design a program and its user interface.  **CRD-2.H:** Acknowledge code segments used from other sources.  **CRD-2.I:** For errors in an algorithm or program:  Identify the error.  Correct the error.  **CRD-2.J:** Identify inputs and corresponding expected outputs or behaviors that can be used to check the correctness of an algorithm or program.  **CRD-2.B.4:** The *behavior* of a program is how a program functions during execution and is often described by how a user interacts with it.  **CRD-2.B.5:** A program can be described broadly by what it does, or in more detail by both what the program does and how the program statements accomplish this function.  **CRD-2.E.1:** A development process can be ordered and intentional, or exploratory in nature.  **CRD-2.E.2:** There are multiple development processes. The following phases are commonly used when developing a program: investigating and reflecting, designing, prototyping, testing  **CRD-2.E.3:** A development process that is iterative requires refinement and revision based on feedback, testing, or reflection throughout the process. This may require revisiting earlier phases of the process. | [AP Performance Tasks Teacher Guide (edc.org)](https://bjc.edc.org/bjc-r/cur/teaching-guide/resources/AP-performance-tasks.html?course=bjc4nyc_teacher.html&novideo&noassignment) | |  |  |
| Nov. 23-27 | **Thanksgiving Break** | | | |  | |  |  |
| December | Lab 1: Computer Networks  (80–160 minutes)  Lab 2: Cybersecurity  (145–290 minutes)  Lab 3: Community and Online Interactions  (185–370 minutes)  Lab 4: Data Representation and Compression  (150–300 minutes) | [**Unit 4: How the Internet Works (edc.org)**](https://bjc.edc.org/bjc-r/topic/topic.html?topic=nyc_bjc/4-internet.topic&course=bjc4nyc.html&novideo&noassignment)  **Lab 1: Computer Networks**  [What Is the Internet?](https://bjc.edc.org/bjc-r/cur/programming/4-internet/1-reliable-communication/1-what-is-internet.html?topic=nyc_bjc%2F4-internet.topic&course=bjc4nyc.html&novideo&noassignment)  [Network Redundancy](https://bjc.edc.org/bjc-r/cur/programming/4-internet/1-reliable-communication/2-network-redundancy.html?topic=nyc_bjc%2F4-internet.topic&course=bjc4nyc.html&novideo&noassignment)  [Internet Abstractions and Open Protocols](https://bjc.edc.org/bjc-r/cur/programming/4-internet/1-reliable-communication/3-open-protocols.html?topic=nyc_bjc%2F4-internet.topic&course=bjc4nyc.html&novideo&noassignment)  [Who's In Charge of the Internet?](https://bjc.edc.org/bjc-r/cur/programming/4-internet/1-reliable-communication/4-who-is-in-charge.html?topic=nyc_bjc%2F4-internet.topic&course=bjc4nyc.html&novideo&noassignment)  **Lab 2: Cybersecurity**  [Cryptography](https://bjc.edc.org/bjc-r/cur/programming/4-internet/2-cybersecurity/1-cryptography.html?topic=nyc_bjc%2F4-internet.topic&course=bjc4nyc.html&novideo&noassignment)  [Caesar Cipher Project](https://bjc.edc.org/bjc-r/cur/programming/4-internet/2-cybersecurity/2-caesar-cipher.html?topic=nyc_bjc%2F4-internet.topic&course=bjc4nyc.html&novideo&noassignment)  [Public Key Encryption](https://bjc.edc.org/bjc-r/cur/programming/4-internet/2-cybersecurity/3-asymmetric_cryptography.html?topic=nyc_bjc%2F4-internet.topic&course=bjc4nyc.html&novideo&noassignment)  [Who Cares About Encryption?](https://bjc.edc.org/bjc-r/cur/programming/4-internet/2-cybersecurity/4-who-cares-about-encryption.html?topic=nyc_bjc%2F4-internet.topic&course=bjc4nyc.html&novideo&noassignment)  [Security Risks](https://bjc.edc.org/bjc-r/cur/programming/4-internet/2-cybersecurity/5-security-risks.html?topic=nyc_bjc%2F4-internet.topic&course=bjc4nyc.html&novideo&noassignment)  [What Can You Do?](https://bjc.edc.org/bjc-r/cur/programming/4-internet/2-cybersecurity/6-cybersecurity.html?topic=nyc_bjc%2F4-internet.topic&course=bjc4nyc.html&novideo&noassignment)  **Lab 3: Community and Online Interactions**  [Communication and Community](https://bjc.edc.org/bjc-r/cur/programming/4-internet/3-community/1-ways-we-communicate.html?topic=nyc_bjc%2F4-internet.topic&course=bjc4nyc.html&novideo&noassignment)  [Cyberbullying](https://bjc.edc.org/bjc-r/cur/programming/4-internet/3-community/2-cyberbullying.html?topic=nyc_bjc%2F4-internet.topic&course=bjc4nyc.html&novideo&noassignment)  [Censorship](https://bjc.edc.org/bjc-r/cur/programming/4-internet/3-community/3-censorship.html?topic=nyc_bjc%2F4-internet.topic&course=bjc4nyc.html&novideo&noassignment)[Search Engines](https://bjc.edc.org/bjc-r/cur/programming/4-internet/3-community/4-search-engines.html?topic=nyc_bjc%2F4-internet.topic&course=bjc4nyc.html&novideo&noassignment)  [Computing Around the World](https://bjc.edc.org/bjc-r/cur/programming/4-internet/3-community/5-computing-world.html?topic=nyc_bjc%2F4-internet.topic&course=bjc4nyc.html&novideo&noassignment)  [Benefits of Computing](https://bjc.edc.org/bjc-r/cur/programming/4-internet/3-community/6-benefits.html?topic=nyc_bjc%2F4-internet.topic&course=bjc4nyc.html&novideo&noassignment)  [Digital Collaboration](https://bjc.edc.org/bjc-r/cur/programming/4-internet/3-community/7-collaboration.html?topic=nyc_bjc%2F4-internet.topic&course=bjc4nyc.html&novideo&noassignment)  [Binary Representation](https://bjc.edc.org/bjc-r/cur/programming/4-internet/4-representation-compression/5-binary.html?topic=nyc_bjc%2F4-internet.topic&course=bjc4nyc.html&novideo&noassignment)  [Data Compression](https://bjc.edc.org/bjc-r/cur/programming/4-internet/4-representation-compression/6-compression.html?topic=nyc_bjc%2F4-internet.topic&course=bjc4nyc.html&novideo&noassignment)  **Lab 4: Data Representation and Compression**  [Bits](https://bjc.edc.org/bjc-r/cur/programming/4-internet/4-representation-compression/1-bits.html?topic=nyc_bjc%2F4-internet.topic&course=bjc4nyc.html&novideo&noassignment)  [Binary Sequences](https://bjc.edc.org/bjc-r/cur/programming/4-internet/4-representation-compression/2-sequences.html?topic=nyc_bjc%2F4-internet.topic&course=bjc4nyc.html&novideo&noassignment)  [Representing Whole Numbers](https://bjc.edc.org/bjc-r/cur/programming/4-internet/4-representation-compression/3-representing-numbers.html?topic=nyc_bjc%2F4-internet.topic&course=bjc4nyc.html&novideo&noassignment)  [Floating Point](https://bjc.edc.org/bjc-r/cur/programming/4-internet/4-representation-compression/4-floating-point.html?topic=nyc_bjc%2F4-internet.topic&course=bjc4nyc.html&novideo&noassignment) | Computer network – computing system – computing device – world wide web – router – internet service providers (ISP’s) – bandwidth – the cloud – path – routing – scalability – redundancy – fault tolerance – protocol – IP address – packet – packet switching – internet protocol (IP) – transmission control protocol (TCP)  Encryption – decryption- symmetric encryption – public key encryption – secure sockets layer/transport layer security (SSL/TLS) – certificate authorities – malware – keylogging software – computer virus – antivirus/anti-malware software – firewall – phishing – DDoS (Disturbed Denial of Service) attack – rogue access point  Digital divide – citizen science – crowdsourcing – computing innovation  Bit – byte – word – binary sequence – analog – sampling- sampling rate – width – floating point – lossless compression – lossy data compression | **CSN-1.A.1:** A *computing device* is a physical artifact that can run a program. Some examples include computers, tablets, servers, routers, and smart sensors.  **CSN-1.A.2:** A *computing system* is a group of computing devices and programs working together for a common purpose.  **CSN-1.A.3:** A *computer network* is a group of interconnected computing devices capable of sending or receiving data.  **CSN-1.A.4:** A computer network is a type of a computing system.  **CSN-1.A.5:** A *path* between two computing devices on a computer network (a sender and a receiver) is a sequence of directly connected computing devices that begins at the sender and ends at the receiver.  **CSN-1.A.6:** *Routing* is the process of finding a path from sender to receiver.  **CSN-1.A.7:** The *bandwidth* of a computer network is the maximum amount of data that can be sent in a fixed amount of time.  **CSN-1.A.8:** Bandwidth is usually measured in bits per second.  **CSN-1.B.1:** The Internet is a computer network consisting of interconnected networks that use standardized, open (nonproprietary) communication protocols.  **CSN-1.B.2:** Access to the Internet depends on the ability to connect a computing device to an Internet-connected device.  **CSN-1.B.3:** A *protocol* is an agreed-upon set of rules that specify the behavior of a system.  **CSN-1.B.4:** The protocols used in the Internet are *open*, which allows users to easily connect additional computing devices to the Internet.  **CSN-1.B.5:** Routing on the Internet is usually dynamic; it is not specified in advance.  **CSN-1.B.6:** The *scalability* of a system is the capacity for the system to change in size and scale to meet new demands.  **CSN-1.B.7:** The Internet was designed to be scalable.  **CSN-1.C.1:** Information is passed through the Internet as a *data stream*. Data streams contain chunks of data, which are encapsulated in *packets*.  **CSN-1.C.2:** Packets contain a chunk of data and metadata used for routing the packet between the origin and the destination on the Internet, as well as for data reassembly.  Fee | [Teaching Guide for Unit 4: How the Internet Works (edc.org)](https://bjc.edc.org/bjc-r/topic/topic.html?topic=nyc_bjc/4-teaching-guide.topic&course=bjc4nyc_teacher.html&novideo&noassignment) | |  |  |
| Dec. 21-Jan 1 | **Christmas Break** | | | |  | |  |  |
| January 2-27 | Lab 1: Search Algorithms and Efficiency  (150–300 minutes) for required pages  Lab 2: Simulations  (40–80 minutes)  Lab 3: Turning Data into Information  (125–250 minutes)  Lab 4: Unsolvable and Undecidable Problems  no required pages  Lab 5: Computing in War  (120–240 minutes)  Lab 6:  Tic-Tac-Toe with a Computer Player | [Unit 5: Algorithms and Simulations (edc.org)](https://bjc.edc.org/bjc-r/topic/topic.html?topic=nyc_bjc/5-algorithms.topic&course=bjc4nyc.html&novideo&noassignment) Lab 1: Search Algorithms and Efficiency [Guess My Number](https://bjc.edc.org/bjc-r/cur/programming/5-algorithms/1-searching-lists/1-sorted-lists.html?topic=nyc_bjc%2F5-algorithms.topic&course=bjc4nyc.html&novideo&noassignment)  [How Many Five-Letter Words Are There?](https://bjc.edc.org/bjc-r/cur/programming/5-algorithms/1-searching-lists/2-how-many-five-letter-words.html?topic=nyc_bjc%2F5-algorithms.topic&course=bjc4nyc.html&novideo&noassignment)  [Is "Seperate" Spelled Correctly?](https://bjc.edc.org/bjc-r/cur/programming/5-algorithms/1-searching-lists/3-spell-checker.html?topic=nyc_bjc%2F5-algorithms.topic&course=bjc4nyc.html&novideo&noassignment)  [Exactly How Much Faster Is Binary Search?](https://bjc.edc.org/bjc-r/cur/programming/5-algorithms/1-searching-lists/4-efficiency.html?topic=nyc_bjc%2F5-algorithms.topic&course=bjc4nyc.html&novideo&noassignment)  [Categorizing Algorithms](https://bjc.edc.org/bjc-r/cur/programming/5-algorithms/1-searching-lists/5-categorizing-algorithms.html?topic=nyc_bjc%2F5-algorithms.topic&course=bjc4nyc.html&novideo&noassignment)  [Heuristic Solutions](https://bjc.edc.org/bjc-r/cur/programming/5-algorithms/1-searching-lists/6-heuristics.html?topic=nyc_bjc%2F5-algorithms.topic&course=bjc4nyc.html&novideo&noassignment)  [Removing Duplicates](https://bjc.edc.org/bjc-r/cur/programming/5-algorithms/1-searching-lists/7-comparing-search-algorithms.html?topic=nyc_bjc%2F5-algorithms.topic&course=bjc4nyc.html&novideo&noassignment)  [Parallelism](https://bjc.edc.org/bjc-r/cur/programming/5-algorithms/1-searching-lists/8-sequential.html?topic=nyc_bjc%2F5-algorithms.topic&course=bjc4nyc.html&novideo&noassignment) Lab 2: Simulations [Why Use Simulations?](https://bjc.edc.org/bjc-r/cur/programming/5-algorithms/2-simulation/1-intro-to-simulations.html?topic=nyc_bjc%2F5-algorithms.topic&course=bjc4nyc.html&novideo&noassignment)  [Disease Spread Project](https://bjc.edc.org/bjc-r/cur/programming/5-algorithms/2-simulation/2-disease-spread.html?topic=nyc_bjc%2F5-algorithms.topic&course=bjc4nyc.html&novideo&noassignment) Lab 3: Turning Data into Information [Analyzing Global Health Data](https://bjc.edc.org/bjc-r/cur/programming/5-algorithms/3-turning-data-information/1-health-data.html?topic=nyc_bjc%2F5-algorithms.topic&course=bjc4nyc.html&novideo&noassignment)  [Self-Check: Big Data](https://bjc.edc.org/bjc-r/cur/programming/5-algorithms/3-turning-data-information/2-self-check.html?topic=nyc_bjc%2F5-algorithms.topic&course=bjc4nyc.html&novideo&noassignment)  [Importing and Accessing Data](https://bjc.edc.org/bjc-r/cur/programming/5-algorithms/3-turning-data-information/3-importing-data.html?topic=nyc_bjc%2F5-algorithms.topic&course=bjc4nyc.html&novideo&noassignment)  [Analyzing Data](https://bjc.edc.org/bjc-r/cur/programming/5-algorithms/3-turning-data-information/4-filtering-data.html?topic=nyc_bjc%2F5-algorithms.topic&course=bjc4nyc.html&novideo&noassignment)  [Visualizing Data](https://bjc.edc.org/bjc-r/cur/programming/5-algorithms/3-turning-data-information/5-visualizing-data.html?topic=nyc_bjc%2F5-algorithms.topic&course=bjc4nyc.html&novideo&noassignment)  [Metadata](https://bjc.edc.org/bjc-r/cur/programming/5-algorithms/3-turning-data-information/6-metadata.html?topic=nyc_bjc%2F5-algorithms.topic&course=bjc4nyc.html&novideo&noassignment) Lab 4: Unsolvable and Undecidable Problems [Proof by Contradiction](https://bjc.edc.org/bjc-r/cur/programming/5-algorithms/4-unsolvable-undecidable/1-logical-inconsistency.html?topic=nyc_bjc%2F5-algorithms.topic&course=bjc4nyc.html&novideo&noassignment)  [An Undecidable Problem](https://bjc.edc.org/bjc-r/cur/programming/5-algorithms/4-unsolvable-undecidable/2-halting-problem.html?topic=nyc_bjc%2F5-algorithms.topic&course=bjc4nyc.html&novideo&noassignment) Lab 5: Computing in War [War and Technology](https://bjc.edc.org/bjc-r/cur/programming/5-algorithms/5-war/1-war-and-technology.html?topic=nyc_bjc%2F5-algorithms.topic&course=bjc4nyc.html&novideo&noassignment)  [Computers and War](https://bjc.edc.org/bjc-r/cur/programming/5-algorithms/5-war/2-machines-and-lives.html?topic=nyc_bjc%2F5-algorithms.topic&course=bjc4nyc.html&novideo&noassignment)  [Military Technology in Everyday Life](https://bjc.edc.org/bjc-r/cur/programming/5-algorithms/5-war/3-everyday-life.html?topic=nyc_bjc%2F5-algorithms.topic&course=bjc4nyc.html&novideo&noassignment) Lab 6: Tic-Tac-Toe with a Computer Player [Review Your Tic-Tac-Toe Project](https://bjc.edc.org/bjc-r/cur/programming/5-algorithms/6-tic-tac-toe/1-review.html?topic=nyc_bjc%2F5-algorithms.topic&course=bjc4nyc.html&novideo&noassignment)  [Creating a Computer Player](https://bjc.edc.org/bjc-r/cur/programming/5-algorithms/6-tic-tac-toe/2-creating-computer-player.html?topic=nyc_bjc%2F5-algorithms.topic&course=bjc4nyc.html&novideo&noassignment)  [Developing a Gameplay Strategy](https://bjc.edc.org/bjc-r/cur/programming/5-algorithms/6-tic-tac-toe/3-developing-strategy.html?topic=nyc_bjc%2F5-algorithms.topic&course=bjc4nyc.html&novideo&noassignment)  [Making the Computer Play Strategically](https://bjc.edc.org/bjc-r/cur/programming/5-algorithms/6-tic-tac-toe/4-making-computer-player-strategic.html?topic=nyc_bjc%2F5-algorithms.topic&course=bjc4nyc.html&novideo&noassignment) | Problem – instance of a problem – linear search – sequential search – linear time – binary search – efficiency – linear time – sublinear time – constant time – quadratic time – polynomial time – exponential time – decision problem – optimization problem – decidable – undecidable – sequential computing – parallel computing – distributed computing – processor – speedup -  Simulations -  Data – information – correlation – insight – record – field – column – cleaning data – classifying data – mode – metadata -  Proof by contradiction – undecidable – self-contradictory – infinite loop – unsolvable problem – undecidable problem - | **AAP-2.O.1:** Traversing a list can be a complete traversal, where all elements in the list are accessed, or a partial traversal, where only a portion of elements are accessed.  **AAP-2.O.5:** Linear search or sequential search algorithms check each element of a list, in order, until the desired value is found or all elements in the list have been checked.  **AAP-2.P.1:** The binary search algorithm starts at the middle of a sorted data set of numbers and eliminates half of the data; this process repeats until the desired value is found or all elements have been eliminated.  **AAP-2.P.2:** Data must be in sorted order to use the binary search algorithm.  **AAP-2.P.3:** Binary search is often more efficient than sequential/linear search when applied to sorted data.  **AAP-4.A.1:** A *problem* is a general description of a task that can (or cannot) be solved algorithmically. An *instance* of a problem also includes specific input. For example, sorting is a problem; sorting the list (2,3,1,7) is an instance of the problem.  **AAP-4.A.2:** A *decision problem* is a problem with a yes/no answer (e.g., is there a path from A to B?). An *optimization problem* is a problem with the goal of finding the "best" solution among many (e.g., what is the shortest path from A to B?).  **AAP-4.A.3:** *Efficiency* is an estimation of the amount of computational resources used by an algorithm. Efficiency is typically expressed as a function of the size of the input.  **AAP-4.A.4:** An algorithm's efficiency is determined through formal or mathematical reasoning.  **AAP-4.A.5:** An algorithm's efficiency can be informally measured by determining the number of times a statement or group of statements executes.  **AAP-4.A.6:** Different correct algorithms for the same problem can have different efficiencies.  **AAP-4.A.7:** Algorithms with a polynomial efficiency or slower (constant, linear, square, cube, etc.) are said to run in a *reasonable amount of time*. Algorithms with exponential or factorial efficiencies are examples of algorithms that run in an *unreasonable amount of time*.  **AAP-4.A.8:** Some problems cannot be solved in a reasonable amount of time because there is no efficient algorithm for solving them. In these cases, approximate solutions are sought.  **AAP-4.A.9:** A *heuristic* is an approach to a problem that produces a solution that is not guaranteed to be optimal but may be used when techniques that are guaranteed to always find an optimal solution are impractical.  **AAP-4.B.1:** A *decidable problem* is a decision problem for which an algorithm can be written to produce a correct output for all inputs (e.g., *Is the number even?*).  **AAP-4.B.2:** An *undecidable problem* is one for which no algorithm can be constructed that is always capable of providing a correct yes-or-no answer.  **AAP-4.B.3:** An undecidable problem may have some instances that have an algorithmic solution, but there is no algorithmic solution that could solve all instances of the problem.  **CSN-2.A.1:** *Sequential computing* is a computational model in which operations are performed in order one at a time.  **CSN-2.A.2:** *Parallel computing* is a computational model where the program is broken into multiple smaller sequential computing operations, some of which are performed simultaneously.  **CSN-2.A.3:** *Distributed computing* is a computational model in which multiple devices are used to run a program.  **CSN-2.A.4:** Comparing efficiency of solutions can be done by comparing the time it takes them to perform the same task.  **CSN-2.A.5:** A sequential solution takes as long as the sum of all of its steps.  **CSN-2.A.6:** A parallel computing solution takes as long as its sequential tasks plus the longest of its parallel tasks. | [Teaching Guide for Unit 5: Algorithms and Data (edc.org)](https://bjc.edc.org/bjc-r/topic/topic.html?topic=nyc_bjc/5-teaching-guide.topic&course=bjc4nyc_teacher.html&novideo&noassignment) | |  |  |
| Jan. 27 |  | **20 WEEK MARKING PERIOD CLOSES** | **20 WEEK MARKING PERIOD CLOSES** | **20 WEEK MARKING PERIOD CLOSES** | **20 WEEK MARKING PERIOD CLOSES** | |  |  |
| Jan.- Feb  30 week marking period | CSP Create Task  (at least 12 hours) | [AP CSP Create Task (edc.org)](https://bjc.edc.org/bjc-r/topic/topic.html?topic=nyc_bjc/create-task.topic&course=bjc4nyc.html&novideo&noassignment)  You will work through the following pages twice: once for the *practice* Create Task (when you can get help from your teacher and classmates) and once for the *official* Create Task (when you can't get help because it counts toward your AP score).  [Using a Development Process to Organize Your Coding](https://bjc.edc.org/bjc-r/cur/performance-tasks/create-task/1-development-process.html?topic=nyc_bjc%2Fcreate-task.topic&course=bjc4nyc.html&novideo&noassignment)  [Choosing Your Project](https://bjc.edc.org/bjc-r/cur/performance-tasks/create-task/2-choosing-your-project.html?topic=nyc_bjc%2Fcreate-task.topic&course=bjc4nyc.html&novideo&noassignment)  [Implementing Your Development Process](https://bjc.edc.org/bjc-r/cur/performance-tasks/create-task/3-implementing.html?topic=nyc_bjc%2Fcreate-task.topic&course=bjc4nyc.html&novideo&noassignment)  [Testing Your Project](https://bjc.edc.org/bjc-r/cur/performance-tasks/create-task/4-testing.html?topic=nyc_bjc%2Fcreate-task.topic&course=bjc4nyc.html&novideo&noassignment)  [Communicating About Your Project](https://bjc.edc.org/bjc-r/cur/performance-tasks/create-task/5-communicating.html?topic=nyc_bjc%2Fcreate-task.topic&course=bjc4nyc.html&novideo&noassignment)  [Evaluating Your Work](https://bjc.edc.org/bjc-r/cur/performance-tasks/create-task/6-evaluating.html?topic=nyc_bjc%2Fcreate-task.topic&course=bjc4nyc.html&novideo&noassignment) | ALL PREVIOUS VOCABULARY | ALL PREVIOUS STANDARDS | [AP Performance Tasks Teacher Guide (edc.org)](https://bjc.edc.org/bjc-r/cur/teaching-guide/resources/AP-performance-tasks.html?course=bjc4nyc_teacher.html&novideo&noassignment)  Teachers must provide **at least 12 classroom hours** to complete the Create Task.  BJC recommends allocating **about 6 classroom hours** for the Practice Create Task. | |  |  |
| March | Lab 1: Computer Abstraction Hierarchy  (180–360 minutes)  Lab 2: History of Computers  (35–70 minutes) | [Unit 6: How Computers Work (edc.org)](https://bjc.edc.org/bjc-r/topic/topic.html?topic=nyc_bjc/6-how-computers-work.topic&course=bjc4nyc.html&novideo&noassignment) Lab 1: Computer Abstraction Hierarchy [Abstraction Inside the Computer](https://bjc.edc.org/bjc-r/cur/programming/6-computers/1-abstraction/01-abstraction.html?topic=nyc_bjc%2F6-how-computers-work.topic&course=bjc4nyc.html&novideo&noassignment)  [The Software Domain: Applications](https://bjc.edc.org/bjc-r/cur/programming/6-computers/1-abstraction/02-software-applications.html?topic=nyc_bjc%2F6-how-computers-work.topic&course=bjc4nyc.html&novideo&noassignment)  [The Software Domain: Programming Languages](https://bjc.edc.org/bjc-r/cur/programming/6-computers/1-abstraction/03-software-languages.html?topic=nyc_bjc%2F6-how-computers-work.topic&course=bjc4nyc.html&novideo&noassignment)  [The Software Domain: Libraries](https://bjc.edc.org/bjc-r/cur/programming/6-computers/1-abstraction/04-software-libraries.html?topic=nyc_bjc%2F6-how-computers-work.topic&course=bjc4nyc.html&novideo&noassignment)  [The Software Domain: Operating Systems](https://bjc.edc.org/bjc-r/cur/programming/6-computers/1-abstraction/05-software-OS.html?topic=nyc_bjc%2F6-how-computers-work.topic&course=bjc4nyc.html&novideo&noassignment)  [The Digital Domain: Architecture](https://bjc.edc.org/bjc-r/cur/programming/6-computers/1-abstraction/06-digital-architecture.html?topic=nyc_bjc%2F6-how-computers-work.topic&course=bjc4nyc.html&novideo&noassignment)  [The Digital Domain: Components](https://bjc.edc.org/bjc-r/cur/programming/6-computers/1-abstraction/07-digital-components.html?topic=nyc_bjc%2F6-how-computers-work.topic&course=bjc4nyc.html&novideo&noassignment)  [The Digital Domain: Integrated Circuits](https://bjc.edc.org/bjc-r/cur/programming/6-computers/1-abstraction/08-digital-IC.html?topic=nyc_bjc%2F6-how-computers-work.topic&course=bjc4nyc.html&novideo&noassignment)  [The Digital Domain: Logic Gates](https://bjc.edc.org/bjc-r/cur/programming/6-computers/1-abstraction/09-digital-logic-gates.html?topic=nyc_bjc%2F6-how-computers-work.topic&course=bjc4nyc.html&novideo&noassignment)  [The Analog Domain: Transistors](https://bjc.edc.org/bjc-r/cur/programming/6-computers/1-abstraction/10-analog.html?topic=nyc_bjc%2F6-how-computers-work.topic&course=bjc4nyc.html&novideo&noassignment) Lab 2: History of Computers [A Brief History of Computers](https://bjc.edc.org/bjc-r/cur/programming/6-computers/2-history-impact/1-timeline.html?topic=nyc_bjc%2F6-how-computers-work.topic&course=bjc4nyc.html&novideo&noassignment)  [Moore's Law](https://bjc.edc.org/bjc-r/cur/programming/6-computers/2-history-impact/2-moore.html?topic=nyc_bjc%2F6-how-computers-work.topic&course=bjc4nyc.html&novideo&noassignment) | Analog- digital – software library – machine language – architecture – integrated circuit  Moore’s Law - | **Learning Goals:**  Understand that the computer abstraction hierarchy includes three major *domains*: software, digital, and analog.  Understand that these domains are separated by the abstraction of a *program* and the abstraction of data being represented *digitally*, as ones and zeros.  Understand that the software and digital domains each include a sub-hierarchy of abstraction.  **Learning Goals:**  Understand Moore's Law, that transistor count in ICs has approximately doubled every year or two.  Understand that computer processor speed and memory size have roughly followed the same growth.  Appreciate that Moore's Law isn't a law of nature and has limitations. | [Teaching Guide for Unit 6: How Computers Work (edc.org)](https://bjc.edc.org/bjc-r/topic/topic.html?topic=nyc_bjc/6-teaching-guide.topic&course=bjc4nyc_teacher.html&novideo&noassignment)  The required content for the AP CS Principles exam is in the first five units. Units 6-8 contain ideas that we consider important, but that are not included in the CS Principles Framework. They are therefore suitable for use in the weeks following the AP exam. | |  |  |
| March 24th |  | **30 WEEK MARKING PERIOD CLOSES** | **30 WEEK MARKING PERIOD CLOSES** | **30 WEEK MARKING PERIOD CLOSES** | **30 WEEK MARKING PERIOD CLOSES** | |  |  |
| 40 weeks |  |  |  |  |  | |  |  |
| April 1-16 | **Spring Break** | | | |  | |  |  |
| April | (at least 12 hours) | AP CSP Create Performance Task | ALL PREVIOUS VOCABUALRY | ALL PREVIOUS STANDARDS | [AP Performance Tasks Teacher Guide (edc.org)](https://bjc.edc.org/bjc-r/cur/teaching-guide/resources/AP-performance-tasks.html?course=bjc4nyc_teacher.html&novideo&noassignment) | |  |  |
| April -May | (at least 12 hours) | Create Task - Written Responses | ALL PREVIOUS VOCABUALRY | ALL PREVIOUS STANDARDS | [AP Performance Tasks Teacher Guide (edc.org)](https://bjc.edc.org/bjc-r/cur/teaching-guide/resources/AP-performance-tasks.html?course=bjc4nyc_teacher.html&novideo&noassignment) | |  |  |
| May 8th 2023 |  |  | AP COMPUTER SCIENCE EXAM |  |  | |  |  |
| May | Lab 1: Trees  (70–140 minutes)  Lab 2: Recursion Projects  (95–190 minutes) for required pages | [Unit 7: Fractals and Recursion (edc.org)](https://bjc.edc.org/bjc-r/topic/topic.html?topic=nyc_bjc/7-recursion-trees-fractals.topic&course=bjc4nyc.html&novideo&noassignment) Lab 1: Trees [Recursive Tree](https://bjc.edc.org/bjc-r/cur/programming/7-recursion/1-trees/1-recursive-tree.html?topic=nyc_bjc%2F7-recursion-trees-fractals.topic&course=bjc4nyc.html&novideo&noassignment)  [The Base Case](https://bjc.edc.org/bjc-r/cur/programming/7-recursion/1-trees/2-base-case.html?topic=nyc_bjc%2F7-recursion-trees-fractals.topic&course=bjc4nyc.html&novideo&noassignment)  [Self-Check: **Tree** Inputs](https://bjc.edc.org/bjc-r/cur/programming/7-recursion/1-trees/3-tree-input-self-check.html?topic=nyc_bjc%2F7-recursion-trees-fractals.topic&course=bjc4nyc.html&novideo&noassignment)  [Self-Check: **Tree** Variations](https://bjc.edc.org/bjc-r/cur/programming/7-recursion/1-trees/4-tree-variations-self-check.html?topic=nyc_bjc%2F7-recursion-trees-fractals.topic&course=bjc4nyc.html&novideo&noassignment)  [Vary Your Tree](https://bjc.edc.org/bjc-r/cur/programming/7-recursion/1-trees/5-vary-your-tree.html?topic=nyc_bjc%2F7-recursion-trees-fractals.topic&course=bjc4nyc.html&novideo&noassignment) Lab 2: Recursion Projects [Triangle Fractal](https://bjc.edc.org/bjc-r/cur/programming/7-recursion/2-projects/1-triangle-fractal.html?topic=nyc_bjc%2F7-recursion-trees-fractals.topic&course=bjc4nyc.html&novideo&noassignment)  [Koch Snowflake](https://bjc.edc.org/bjc-r/cur/programming/7-recursion/2-projects/2-snowflake.html?topic=nyc_bjc%2F7-recursion-trees-fractals.topic&course=bjc4nyc.html&novideo&noassignment)  [Lévy C-Curve Fractal](https://bjc.edc.org/bjc-r/cur/programming/7-recursion/2-projects/3-c-curve.html?topic=nyc_bjc%2F7-recursion-trees-fractals.topic&course=bjc4nyc.html&novideo&noassignment)  [Recursive Mondrian](https://bjc.edc.org/bjc-r/cur/programming/7-recursion/2-projects/4-mondrian.html?topic=nyc_bjc%2F7-recursion-trees-fractals.topic&course=bjc4nyc.html&novideo&noassignment) | Fractal – state transparency – recursion – base case | **Learning Goal:**  Get a collective introduction to recursion through the **Vee** project, which selects blocks to draw from a list of shape blocks that includes the **vee** block itself.  **Learning Goal:**  Practice recursion by developing several fractal-drawing programs.  Apply mathematics (including Pythagorean formula, coordinate positioning, and angles) to computer graphics. | [Teaching Guide for Unit 7: Fractals and Recursion (edc.org)](https://bjc.edc.org/bjc-r/topic/topic.html?topic=nyc_bjc/7-teaching-guide.topic&course=bjc4nyc_teacher.html&novideo&noassignment) | |  |  |
| June | Lab 1: Recursive Reporters  (80–160 minutes)  Lab 2: Base Conversion  (45–90 minutes)  Lab 3: Subsets  (70–140 minutes)  Lab 4: Building Higher Order Functions  (80–160 minutes) | [Unit 8: Recursive Functions (edc.org)](https://bjc.edc.org/bjc-r/topic/topic.html?topic=nyc_bjc/8-recursive-reporters.topic&course=bjc4nyc.html&novideo&noassignment) Lab 1: Recursive Reporters [Counting Trees](https://bjc.edc.org/bjc-r/cur/programming/8-recursive-reporters/1-recursive-reporters/1-counting-trees.html?topic=nyc_bjc%2F8-recursive-reporters.topic&course=bjc4nyc.html&novideo&noassignment)  [Writing Recursive Reporters](https://bjc.edc.org/bjc-r/cur/programming/8-recursive-reporters/1-recursive-reporters/2-writing-recursive-reporters.html?topic=nyc_bjc%2F8-recursive-reporters.topic&course=bjc4nyc.html&novideo&noassignment) Lab 2: Base Conversion [Binary Conversion](https://bjc.edc.org/bjc-r/cur/programming/8-recursive-reporters/2-base-conversion/1-binary-conversion.html?topic=nyc_bjc%2F8-recursive-reporters.topic&course=bjc4nyc.html&novideo&noassignment)  [Base Conversion](https://bjc.edc.org/bjc-r/cur/programming/8-recursive-reporters/2-base-conversion/2-base-conversion.html?topic=nyc_bjc%2F8-recursive-reporters.topic&course=bjc4nyc.html&novideo&noassignment) Lab 3: Subsets [Counting Ice Cream Bowls](https://bjc.edc.org/bjc-r/cur/programming/8-recursive-reporters/3-subsets/1-ice-cream-bowls.html?topic=nyc_bjc%2F8-recursive-reporters.topic&course=bjc4nyc.html&novideo&noassignment)  [Listing the Subsets](https://bjc.edc.org/bjc-r/cur/programming/8-recursive-reporters/3-subsets/2-subsets.html?topic=nyc_bjc%2F8-recursive-reporters.topic&course=bjc4nyc.html&novideo&noassignment) Lab 4: Building Higher Order Functions [Walking Down a List](https://bjc.edc.org/bjc-r/cur/programming/8-recursive-reporters/4-building-higher-order-functions/1-plurals-and-other-recursions.html?topic=nyc_bjc%2F8-recursive-reporters.topic&course=bjc4nyc.html&novideo&noassignment)  [Generalizing the **Map** Pattern](https://bjc.edc.org/bjc-r/cur/programming/8-recursive-reporters/4-building-higher-order-functions/2-generalizing-the-map-pattern.html?topic=nyc_bjc%2F8-recursive-reporters.topic&course=bjc4nyc.html&novideo&noassignment)  [Even Numbers and **Keep**](https://bjc.edc.org/bjc-r/cur/programming/8-recursive-reporters/4-building-higher-order-functions/3-even-number-filter.html?topic=nyc_bjc%2F8-recursive-reporters.topic&course=bjc4nyc.html&novideo&noassignment) | subset | **Learning Goal:**  Understand how recursive reporters differ from recursive commands.  Understand combiners in recursive reporters.  Practice recursive functions.  Learn a more elegant approach to base conversion.  Generalize from binary to base conversion in general.  Explore a problem with a solution of exponentially increasing length.  Explore one of the classic problems in computer science: subsets.  Investigate the code pattern that will be generalized into the **map** function  Learn how to write the higher-order function **map**.  Get to know problems for which you can't use **map** itself but for which you can fall back to the **map** code pattern.  Build the higher-order functions **keep** and **combine**. | [Teaching Guide for Unit 8: Recursive Functions (edc.org)](https://bjc.edc.org/bjc-r/topic/topic.html?topic=nyc_bjc/8-teaching-guide.topic&course=bjc4nyc_teacher.html&novideo&noassignment) | |  |  |
| June 16 marking period closes |  | **40 WEEK MARKING PERIOD CLOSES** | **40 WEEK MARKING PERIOD CLOSES** | **40 WEEK MARKING PERIOD CLOSES** | **40 WEEK MARKING PERIOD CLOSES** | |  |  |