FREEHOLD REGIONAL HIGH SCHOOL DISTRICT

Board of Education

Mr. Ronald G. Lawson, President
Mr. Heshy Moses, Vice President

Mr. William Bruno
Mr. Tom Caiazza
Mrs. Elizabeth Canario
Mr. Barry Hochberg
Mrs. Kathie Lavin
Mr. Christopher Placitella
Mrs. Jennifer Sutera

Dr. Suzanne Koegler, Superintendent
Ms. Donna M. Evangelista, Assistant Superintendent for Curriculum and Instruction

Curriculum Writing Committee

Mr. Erik Levin

Supervisor

Ms. Denise Scanga
Course Philosophy

The philosophy of the Science/Engineering AP Computer Science course is to:

- Meet the requirements of the College Board as indicated by successful performance on the Advanced Placement Examination.
- Understand and appreciate the designated advanced placement programming language and special applications of the data structures supported by the language and demonstrate by completion of these applications and projects.
- Acquire familiarity with the algorithms that are used to manipulate the data, procedures, and objects in standard problems encountered in fields such as data processing, mathematical manipulation, and science applications as demonstrated by solving such problems.
- Know and use the criteria that should be employed in order to effectively select and appropriately use in solving problems.
- Know and adhere to the rules of computer etiquette and ethics as evidenced by appropriate and courteous behavior and ethics.

Course Description

Students will complete the requirements of the Advanced Placement Computer Science course, as well as examining and implementing various computer applications and independent projects. Throughout the course, interdisciplinary units will enable students to apply their newly acquired sophisticated programming techniques to practical situations. The culminating project is a major component assigned to teams of students with the explicit purpose of reinforcing program design, style, and algorithm selection.

Students enrolled in this course will demonstrate mastery of the following proficiency requirements:

The student is expected to understand:

- How object oriented design leads to an understandable, reusable and maintainable program.
- How interfaces and classes lead to reusable and maintainable programs.
- That object oriented design is an important part of modern program implementation.
- That various tools and methods are appropriate for testing and validating software.
- Design methods for reusable code and ways of adapting code for reuse.
- Runtime analysis and the influence of the underlying representations on performance and accuracy in calculation.
- Design and analysis of the underlying representations of abstract data and tools for manipulating data in complex ways.
- Standard algorithms and processes applied to standard structures.
- The syntax of C++, JAVA, or other appropriate language as specified by the advanced placement curriculum.
- Computer etiquette and ethics.
- Experiment with alternative programming models and complete small group programming projects.
<table>
<thead>
<tr>
<th>Relevant Standards</th>
<th>Enduring Understandings</th>
<th>Essential Questions</th>
<th>Assessments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit #1: Object Oriented Design</td>
<td>Object Oriented Design can lead to an understandable, reusable, and maintainable programs</td>
<td>How does one read, understand, and specify a problem description? How and why does one abstract and encapsulate data? What are “is-a” and “has-a” relationships in class specification? What are reusable components? How are class relationships defined? How are reusable components from existing class libraries re-used?</td>
<td>Oral discussion</td>
</tr>
<tr>
<td>Unit #2: Object Oriented Design, Part 2</td>
<td>Interfaces and classes can lead to reusable and maintainable programs</td>
<td>What is the difference between an interface and a class? Why does an interface not include implementation of methods? What is the relationship between the interface specification and the eventual implementation via inheritance?</td>
<td>Oral discussion</td>
</tr>
<tr>
<td>Unit #3: Program Implementation</td>
<td>Object oriented design is an important model for program implementation and data representation</td>
<td>How does top down development relate to encapsulation, abstraction, and object oriented development? How do systems evolve from program model via individual classes and segments? Why are objects used to encapsulate primitive data and methods?</td>
<td>Oral discussion</td>
</tr>
<tr>
<td>Unit #4: Program Analysis</td>
<td>Appropriate design methods allow for writing correct code and reusing code can help with correctness</td>
<td>When is it appropriate to test a class in isolation? What are the characteristics of boundary cases and how may they be tested? How are errors identified and addressed?</td>
<td>Oral discussion</td>
</tr>
<tr>
<td>Unit #5: Code Reuse</td>
<td>Appropriate design methods allow for reusable code and ease adapting code for reuse</td>
<td>When is it appropriate to modify existing code for new use? How is inheritance used to extend existing code? How can pre- and post-conditions and assertions be used to ease the reuse process? Why is the exception model used and how does it ease</td>
<td>Oral discussion</td>
</tr>
<tr>
<td>Relevant Standards</td>
<td>Enduring Understandings</td>
<td>Essential Questions</td>
<td>Assessments</td>
</tr>
<tr>
<td>--------------------</td>
<td>-------------------------</td>
<td>---------------------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| **Unit #6: Algorithms I**  
NJCCCS 4.5  
D1, 3, 4;  
E1, 3  
8.2 B  
9.4.12.K(4) | Underlying representations and methods influence performance and accuracy of numerical computation | How are estimates of run time and space needs determined and described?  
How are numbers represented and interpreted by the computer? | Oral discussion  
Problem solving tasks  
Practice Assignments | Oral presentation  
Lab assignments and projects  
Research assignments | Projects  
Midyear and Final exams  
Standardized exams |
| **Unit #7: Algorithms II**  
NJCCCS 4.5  
D1, 3, 4;  
E1, 3  
8.2 B  
9.4.12.K(4) | Design and analysis of underlying representations of abstract data and tools allow us to manipulate data in complex ways | How are estimates of run time and space needs determined and described?  
How are abstract and concrete data represented and interpreted by the computer? | Oral discussion  
Problem solving tasks  
Practice Assignments | Oral presentation  
Lab assignments and projects  
Research assignments | Projects  
Midyear and Final exams  
Standardized exams |
| **Unit #8: Standard Algorithms**  
NJCCCS 4.5  
D1, 3, 4;  
E1, 3  
8.2 B  
9.4.12.K(4) | Standard algorithms, processes, and structures ease the implementation of data processing solutions | How may a structure be traversed?  
How may a structure be used to perform a useful task?  
How do you choose the correct tools for a given task? | Oral discussion  
Problem solving tasks  
Practice Assignments | Oral presentation  
Lab assignments and projects  
Research assignments | Projects  
Midyear and Final exams  
Standardized exams |
<table>
<thead>
<tr>
<th>Unit Title</th>
<th>Unit Understandings and Goals</th>
<th>Recommended Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit #1: Object Oriented Design</td>
<td>Object Oriented Design can lead to an understandable, reusable, and maintainable program. 1. Students will read and understand a problem description. 2. Students will specify purpose and goals for a problem. 3. Students will apply data abstraction and encapsulation concepts. 4. Students will read class specifications and decompose problems into classes. 5. Students will implement class hierarchies. 6. Students will identify reusable class components. 7. Students will choose appropriate data structures.</td>
<td>4 weeks</td>
</tr>
<tr>
<td>Unit #2: Object Oriented Design, Part 2</td>
<td>Interfaces and classes can lead to reusable and maintainable programs. 1. Students will specify an interface. 2. Students will implement an interface. 3. Students will read and use abstract classes.</td>
<td>4 weeks</td>
</tr>
<tr>
<td>Unit #3: Program Implementation</td>
<td>Object oriented design is an important model for program implementation and data representation. 1. Students will declare constants and variables. 2. Students will declare classes, interfaces, and methods. 3. Students will use parameters. 4. Students will perform console output. 5. Students will use standard control structures for iteration, recursion, and conditional statements. 6. Students will use methods from standard classes.</td>
<td>4 weeks</td>
</tr>
<tr>
<td>Unit #4: Program Analysis</td>
<td>Appropriate design methods allow for writing correct code, and reusing code can help with correctness. 1. Students will hand trace code. 2. Students will identify and correct errors. 3. Students will use and design test data suites.</td>
<td>5 weeks</td>
</tr>
<tr>
<td>Unit #5: Code Reuse</td>
<td>Appropriate design methods allow for reusable code and ease adapting code for reuse. 1. Students will throw and handle runtime exceptions. 2. Students will extend existing code using inheritance. 3. Students will interpret and modify existing classes and methods. 4. Students will interpret preconditions, post conditions, and invariants. 5. Students will write code to meet preconditions, post conditions, and invariants.</td>
<td>4 weeks</td>
</tr>
<tr>
<td>Unit #6: Algorithms I</td>
<td>Underlying representations and methods influence performance and accuracy of numerical computation. 1. Students will estimate and model runtime. 2. Students will estimate and model space usage. 3. Students will understand and evaluate error during numerical computation.</td>
<td>3 weeks</td>
</tr>
</tbody>
</table>
| Unit #7: Algorithms II  
| NJCCCS 4.5 D1, 3, 4;  
| E1, 3  
| 8.2 B  
| 9.4.12.K(4)  |
|---|---|
| Design and analysis of underlying representations of abstract data and tools allow us to manipulate data in complex ways | 5 weeks |
| 1. Students will estimate and model runtime.  
| 2. Students will estimate and model space usage.  
| 3. Students will understand how data context and use influences appropriate representation. | |

| Unit #8: Standard Algorithms  
| NJCCCS 4.5 D1, 3, 4;  
| E1, 3  
| 8.2 B  
| 9.4.12.K(4)  |
|---|---|
| Standard algorithms and processes and standard structures ease the implementation of data processing solutions | 3 weeks |
| 1. Students will estimate and model runtime.  
| 2. Students will estimate and model space usage.  
| 3. Students will implement standard algorithms.  
| 4. Students will use standard algorithms to solve common problems. | |
Freehold Regional High School District  
Science and Engineering Advanced Placement Computer Science A  
Unit #1: Object Oriented Design

**Enduring Understanding:** Object Oriented Design can lead to an understandable, reusable, and maintainable program.

**Essential Questions:**
- How does one read, understand, and specify a problem description?
- How and why does one abstract and encapsulate data?
- What are “is-a” and “has-a” relationships in class specification?
- What are reusable components?
- How are class relationships defined?
- How are reusable components from existing class libraries re-used?

**Unit Goals:** Students will read and understand a problem description.  
Students will specify purpose and goals for a problem.  
Students will apply data abstraction and encapsulation concepts.  
Students will read class specifications and decompose problems into classes.  
Students will implement class hierarchies.  
Students will identify reusable class components.  
Students will choose appropriate data structures.

**Duration of Unit:** 4 weeks

**NJCCS:** 4.5 E2, E3; 8.1 A, E; 8.2 B; 9.4.12.K (4)

<table>
<thead>
<tr>
<th>Guiding / Topical Questions</th>
<th>Content, Themes, Concepts, and Skills</th>
<th>Instructional Resources and Materials</th>
<th>Teaching Strategies</th>
<th>Assessment Strategies</th>
</tr>
</thead>
<tbody>
<tr>
<td>How are problems specified?</td>
<td>Read and understand a problem description or operational specification</td>
<td>Textbook</td>
<td>Lecture</td>
<td>Projects</td>
</tr>
<tr>
<td>How do we set solution goals?</td>
<td>Specify purpose and requirements for a programmatic solution</td>
<td>Lecture</td>
<td>Examples</td>
<td>Guided practice problems</td>
</tr>
<tr>
<td>How and why do we encapsulate data?</td>
<td>Application of data abstraction and encapsulation</td>
<td>Computer equipment and appropriate software</td>
<td>Guided practice problems</td>
<td>Test/quizzes</td>
</tr>
<tr>
<td>How do we specify a class?</td>
<td>Specify the data, methods, and modes of access for classes</td>
<td>Sample programs</td>
<td>Student presentation</td>
<td></td>
</tr>
<tr>
<td>How do we decompose problems into classes and class hierarchies?</td>
<td>Specify and use class hierarchies to solve programming problems</td>
<td>Standard documentation</td>
<td>Co-operative learning</td>
<td></td>
</tr>
<tr>
<td>What can be reused?</td>
<td>Design and identify reusable classes and class components</td>
<td>Articles</td>
<td>Classroom discussion</td>
<td></td>
</tr>
</tbody>
</table>

Projects should include the use of standard wrapper classes, the Math class, the String class, appropriate methods from these classes, the use of public and private data and methods, the use of inherited methods, such as to String() and equals().

**Suggestions on how to differentiate in this unit:**
- The use of cooperative learning, assessment that includes projects and presentations, along with the use of technology, will meet the needs of any diverse learner.
Enduring Understanding: Interfaces and classes can lead to reusable and maintainable programs

Essential Questions:
- What is the difference between an interface and a class?
- Why does an interface not include implementation of methods?
- What is the relationship between the interface specification and the eventual implementation via inheritance?

Unit Goals:
- Students will specify an interface.
- Students will implement an interface.
- Students will read and use abstract classes.

Duration of Unit: 4 weeks

NJCCCS: 4.5 E2, E3; 8.1 E; 8.2 B; 9.4.12.K (4)

<table>
<thead>
<tr>
<th>Guiding / Topical Questions</th>
<th>Content, Themes, Concepts, and Skills</th>
<th>Instructional Resources and Materials</th>
<th>Teaching Strategies</th>
<th>Assessment Strategies</th>
</tr>
</thead>
<tbody>
<tr>
<td>What is an interface?</td>
<td>Understand some standard interfaces</td>
<td>Textbook</td>
<td>Lecture</td>
<td>Projects</td>
</tr>
<tr>
<td>Why do we use interfaces?</td>
<td>Apply some standard interfaces</td>
<td>Lecture</td>
<td>Examples</td>
<td>Guided practice problems</td>
</tr>
<tr>
<td>How is an interface designed and used?</td>
<td>Develop classes that implement standard interfaces</td>
<td>Computer equipment and appropriate software</td>
<td>Guided practice problems</td>
<td>Test/quizzes</td>
</tr>
<tr>
<td>What is an abstract class?</td>
<td>Read and use abstract classes</td>
<td>Sample programs</td>
<td>Student presentation</td>
<td>Coöperative learning</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Standard documentation</td>
<td>Classroom discussion</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Articles</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Internet resources</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Projects should include the use of Comparable interface, List interface, and the interfaces for iterating a structure.

Suggestions on how to differentiate in this unit:
- The use of cooperative learning, assessment that includes projects and presentations, along with the use of technology, will meet the needs of any diverse learner.
Enduring Understanding:  Object oriented design is an important model for program implementation and data representation.

Essential Questions:  How does top down development relate to encapsulation, abstraction, and object oriented development?
How do systems evolve from program model via individual classes and segments?
Why are objects used to encapsulate primitive data and methods?

Unit Goals:  Students will declare constants and variables.
Students will declare classes, interfaces, and methods.
Students will use parameters.
Students will perform console output.
Students will use standard control structures for iteration, recursion, and conditional statements.
Students will use methods from standard classes.

Duration of Unit:  4 weeks
NJCCCS:  4.5 E2, E3; 8.1 E; 8.2 B, E, F; 9.4.12.K (4)

<table>
<thead>
<tr>
<th>Guiding / Topical Questions</th>
<th>Content, Themes, Concepts, and Skills</th>
<th>Instructional Resources and Materials</th>
<th>Teaching Strategies</th>
<th>Assessment Strategies</th>
</tr>
</thead>
<tbody>
<tr>
<td>What are elementary data types?</td>
<td>Develop code that uses elementary types</td>
<td>Textbook</td>
<td>Lecture</td>
<td>Projects</td>
</tr>
<tr>
<td>How can data access be specified?</td>
<td>Use public and private data, mutable and immutable data</td>
<td>Lecture</td>
<td>Examples</td>
<td>Guided practice problems</td>
</tr>
<tr>
<td>How are objects different than elementary data?</td>
<td>Use objects of standard classes and understand access methods for objects and member data</td>
<td>Computer equipment and appropriate software</td>
<td>Guided practice problems</td>
<td>Test/quizzes</td>
</tr>
<tr>
<td>How are methods given information?</td>
<td>Use explicit and implicit parameters</td>
<td>Sample programs</td>
<td>Student presentation</td>
<td></td>
</tr>
<tr>
<td>What are standard control structures?</td>
<td>Use standard looping and conditional structures</td>
<td>Standard documentation</td>
<td>Coöperative learning</td>
<td></td>
</tr>
<tr>
<td>What are standard motifs for working with complex structures?</td>
<td>Use the standard iteration model</td>
<td>Articles</td>
<td>Classroom discussion</td>
<td></td>
</tr>
</tbody>
</table>

Projects should include the use of elementary data types, public and private data, operations on these data types, use of the iteration model and the for-each construct, appropriate use of casts, the use of standard wrapper classes, use of standard arrays, and the use of standard classes such as ArrayList and String.

Suggestions on how to differentiate in this unit:
- The use of cooperative learning, assessment that includes projects and presentations, along with the use of technology, will meet the needs of any diverse learner.
Enduring Understanding:  Appropriate design methods allow for writing correct code, and reusing code can help with correctness.

Essential Questions:  
- When is it appropriate to test a class in isolation?
- What are the characteristics of boundary cases how may they be tested?
- How are errors identified and addressed?

Unit Goals:  Students will hand trace code.
- Students will identify and correct errors.
- Students will use and design test data suites.

Duration of Unit:  5 weeks

NJCCCS:  4.5 A1, 3; B2, 3; D2, 4, 6; 8.2 D; 9.4.12.K (4)

<table>
<thead>
<tr>
<th>Guiding / Topical Questions</th>
<th>Content, Themes, Concepts, and Skills</th>
<th>Instructional Resources and Materials</th>
<th>Teaching Strategies</th>
<th>Assessment Strategies</th>
</tr>
</thead>
<tbody>
<tr>
<td>How do we determine the purpose of existing code?</td>
<td>Hand tracing code to determine operation; Use of standard documentation models</td>
<td>Textbook</td>
<td>Lecture</td>
<td>Projects</td>
</tr>
<tr>
<td>How may errors be identified?</td>
<td>Generation and use of test cases; identifying common error syndromes; Locating logical/structural errors through isolation; Use scaffolding to test classes and methods in isolation</td>
<td>Lecture</td>
<td>Examples</td>
<td>Guided practice problems</td>
</tr>
<tr>
<td>What language features help prevent common errors?</td>
<td>Use generic classes to detect and help prevent use of incompatible types</td>
<td>Computer equipment and appropriate software</td>
<td>Guided practice problems</td>
<td>Test/quizzes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sample programs</td>
<td>Student presentation</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Standard documentation</td>
<td>Coöperative learning</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Articles</td>
<td>Classroom discussion</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Internet resources</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Projects should involve use of appropriate documentation as an aid to avoiding and correcting errors

Suggestions on how to differentiate in this unit:
- The use of cooperative learning, assessment that includes projects and presentations, along with the use of technology, will meet the needs of any diverse learner.
Freehold Regional High School District
Science and Engineering Advanced Placement Computer Science A
Unit #5: Code Reuse

Enduring Understanding: Appropriate design methods allow for reusable code and ease adapting code for reuse.

Essential Questions: When is it appropriate to modify existing code for new use?
How is inheritance used to extend existing code?
How can pre- and post-conditions and assertions be used to ease the reuse process?
Why is the exception model used and how does it ease code reuse?

Unit Goals: Students will throw and handle runtime exceptions.
Students will extend existing code using inheritance.
Students will interpret and modify existing classes and methods.
Students will interpret preconditions, post conditions, and invariants.
Students will write code to meet preconditions, post conditions, and invariants.

Duration of Unit: 4 weeks
NJCCCS: 4.5 A3; B; C2; 9.4.12.K (4)

<table>
<thead>
<tr>
<th>Guiding / Topical Questions</th>
<th>Content, Themes, Concepts, and Skills</th>
<th>Instructional Resources and Materials</th>
<th>Teaching Strategies</th>
<th>Assessment Strategies</th>
</tr>
</thead>
<tbody>
<tr>
<td>What properties make code reusable?</td>
<td>Use and develop formal interfaces for methods and classes</td>
<td>Textbook</td>
<td>Lecture</td>
<td>Projects</td>
</tr>
<tr>
<td>What tools help make code reusable?</td>
<td>Use standard exception handling interface; Use standard documentation methods for interfaces;</td>
<td>Lecture</td>
<td>Examples</td>
<td>Guided practice problems</td>
</tr>
<tr>
<td>What is inheritance?</td>
<td>Develop classes that inherit and use properties of the parent class; Use overloading of methods to modify properties</td>
<td>Computer equipment and appropriate software</td>
<td>Guided practice problems</td>
<td>Test/quizzes</td>
</tr>
<tr>
<td>How do we document to ease reuse?</td>
<td>Use preconditions, post conditions, invariants, and interface specifications</td>
<td>Sample programs</td>
<td>Student presentation</td>
<td></td>
</tr>
<tr>
<td>What properties are shared by all objects?</td>
<td>Understand that there is a root to the class hierarchy; explain why certain methods exist for all classes; use, extend, and overload these methods</td>
<td>Standard documentation</td>
<td>Co-operative learning</td>
<td></td>
</tr>
</tbody>
</table>

Projects should involve meeting the contract requirements for standard interfaces and the use of appropriate documentation to allow for the extension and reuse of code. They should also involve the hiding class methods and overriding of instance methods. Projects should include Exceptions being thrown and caught.

Suggestions on how to differentiate in this unit:
- The use of cooperative learning, assessment that includes projects and presentations, along with the use of technology, will meet the needs of any diverse learner.
Enduring Understanding: Underlying representations and methods influence performance and accuracy of numerical computation.

Essential Questions: How are estimates of run time and space needs determined and described?

How are numbers represented and interpreted by the computer?

Unit Goals: Students will estimate and model runtime.

Students will estimate and model space usage.

Students will understand and evaluate error during numerical computation.

Duration of Unit: 3 weeks

NJCCCS: 4.5 D1, 3, 4; E1, 3; 8.2 B; 9.4.12.K (4)

<table>
<thead>
<tr>
<th>Guiding / Topical Questions</th>
<th>Content, Themes, Concepts, and Skills</th>
<th>Instructional Resources and Materials</th>
<th>Teaching Strategies</th>
<th>Assessment Strategies</th>
</tr>
</thead>
<tbody>
<tr>
<td>What are common numerical methods?</td>
<td>Write programs to perform common computational tasks</td>
<td>Textbook</td>
<td>Lecture</td>
<td>Projects</td>
</tr>
<tr>
<td>What are common non-numeric algorithms?</td>
<td>Implement common sorts, searches, and other common methods</td>
<td>Lecture</td>
<td>Examples</td>
<td>Guided practice problems</td>
</tr>
<tr>
<td>How can run time be estimated? How can space use be estimated?</td>
<td>Use ‘Big O’ notation to represent the growth behavior of time and resource usage</td>
<td>Computer equipment and appropriate software</td>
<td>Guided practice problems</td>
<td>Test/quizzes</td>
</tr>
<tr>
<td>What limits correctness of results?</td>
<td>Examine common ways systematic computational error is manifested; Examine how details of implementation interact with representation to improve or reduce correctness</td>
<td>Sample programs</td>
<td>Student presentation</td>
<td>Projects</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Standard documentation</td>
<td>Coöperative learning</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Articles</td>
<td>Classroom discussion</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Internet resources</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Projects should include the implementation of common numerical tools, sorts including merge sort, insertion sort, selection sort, and quick sort.

Suggestions on how to differentiate in this unit:

- The use of cooperative learning, assessment that includes projects and presentations, along with the use of technology, will meet the needs of any diverse learner.
### Freehold Regional High School District
### Science and Engineering Advanced Placement Computer Science A
### Unit #7: Algorithms II

**Enduring Understanding:** Design and analysis of underlying representations of abstract data and tools allow us to manipulate data in complex ways.

**Essential Questions:**
- How are estimates of run time and space needs determined and described?
- How are abstract and concrete data represented and interpreted by the computer?

**Unit Goals:**
- Students will estimate and model runtime
- Students will estimate and model space usage
- Students will understand how data context and use influences appropriate representation

**Duration of Unit:** 5 weeks

**NJCCCS:** 4.5 D1, 3, 4; E1, 3; 8.2 B; 9.4.12.K (4)

<table>
<thead>
<tr>
<th>Guiding / Topical Questions</th>
<th>Content, Themes, Concepts, and Skills</th>
<th>Instructional Resources and Materials</th>
<th>Teaching Strategies</th>
<th>Assessment Strategies</th>
</tr>
</thead>
<tbody>
<tr>
<td>What are some common data structures?</td>
<td>Implement common structures such as lists, arrays, hash-tables, trees, queues, and stacks</td>
<td>Textbook</td>
<td>Lecture</td>
<td>Projects</td>
</tr>
<tr>
<td>What are some common algorithms that act on data structures?</td>
<td>Implement tools such as searches and traversals</td>
<td>Lecture</td>
<td>Examples</td>
<td>Guided practice problems</td>
</tr>
<tr>
<td>How do we analyze the time and space required by a structure and operations on it?</td>
<td>Analyze properties of structures and determine time and space behaviors</td>
<td>Computer equipment and appropriate software</td>
<td>Guided practice problems</td>
<td>Test/quizzes</td>
</tr>
<tr>
<td>How do we choose appropriate structures?</td>
<td>Examine common applications; Understand how the properties of a structure influence selection for a task</td>
<td>Sample programs, Standard documentation, Articles, Internet resources</td>
<td>Student presentation, Coöperative learning</td>
<td></td>
</tr>
</tbody>
</table>

Projects should include the implementation and application of structures such as linked lists, queues, stacks, multidimensional arrays, and the use of common generic standard classes.

**Suggestions on how to differentiate in this unit:**
- The use of cooperative learning, assessment that includes projects and presentations, along with the use of technology, will meet the needs of any diverse learner.
Enduring Understanding: Standard algorithms and processes and standard structures ease the implementation of data processing solutions.

Essential Questions:
- How may a structure be traversed?
- How may a structure be used to perform a useful task?
- How do you choose the correct tools for a given task?

Unit Goals:
- Students will estimate and model runtime.
- Students will estimate and model space usage.
- Students will implement standard algorithms.
- Students will use standard algorithms to solve common problems.

Duration of Unit: 3 weeks

NJCCCS: 4.5 D1, 3, 4; E1, 3; 8.2 B; 9.4.12.K (4)

<table>
<thead>
<tr>
<th>Guiding / Topical Questions</th>
<th>Content, Themes, Concepts, and Skills</th>
<th>Instructional Resources and Materials</th>
<th>Teaching Strategies</th>
<th>Assessment Strategies</th>
</tr>
</thead>
<tbody>
<tr>
<td>What is traversal?</td>
<td>Implement traversals for common structures</td>
<td>Textbook</td>
<td>Lecture</td>
<td>Projects</td>
</tr>
<tr>
<td>What are applications for traversal?</td>
<td>Use traversal for such tasks as searching, parsing, etc</td>
<td>Lecture</td>
<td>Examples</td>
<td>Guided practice problems</td>
</tr>
<tr>
<td>How do we select a particular traversal for a structure?</td>
<td>Examine structures with multiple 'natural' traversals, such as binary trees, and applications of various traversals</td>
<td>Computer equipment and appropriate software</td>
<td>Student presentation</td>
<td>Test/quizzes</td>
</tr>
<tr>
<td>What is sorting?</td>
<td>Implement, compare, and contrast common sorting methods</td>
<td>Sample programs</td>
<td>Coöperative learning</td>
<td></td>
</tr>
<tr>
<td>What is searching?</td>
<td>Implement, compare, and contrast common searching methods for common structures</td>
<td>Standard documentation</td>
<td>Classroom discussion</td>
<td></td>
</tr>
<tr>
<td>Where do we use lists?</td>
<td>Examine applications where list properties are appropriate</td>
<td>Articles</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Where do we use queues and stacks?</td>
<td>Examine algorithms where stacks and queues are appropriate</td>
<td>Internet resources</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Where do we use other common structures?</td>
<td>Examine algorithms that use structures such as trees, hash tables, etc.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Projects should include the use of common search modes such as sequential search and binary search, the use of traversals (including the standard iterator model) in searching, the application and implementation of 'natural' traversals on appropriate structures (such as the use of sequential search on a linked list,) and applications of stacks and queues to searching, traversing, and common tasks such as parsing, evaluation, and elimination of recursion.

Suggestions on how to differentiate in this unit:
- The use of cooperative learning, assessment that includes projects and presentations, along with the use of technology, will meet the needs of any diverse learner.