East Penn School District
Secondary Curriculum

A Planned Course Statement for Advanced Data Structures

Course #367 **Grade(s) 11-12**

Department: Computer Science

Length of Period (mins.) 41 **Total Clock Hours** 63

Periods per Week 5 **Length of Course (yrs.)** .5

Type of Offering: ____ **required** x **elective**

CREDIT .5

Adopted 4/27/09

Developed by:
Carlen Blackstone
Beth Stoudt

Description of Course #370
Course Title: Advanced Data Structures (AP Weight)

Description: This course will build on a solid foundation of computing methodology to introduce students to advanced representation and processing of data. Topics will include algorithm efficiency, recursion, inheritance, and dynamic memory allocation. Students will learn how to process data that is stored as strings, arrays, stacks, queues, linked lists, sets, maps, files, and tree structures to solve a variety of real life application problems.

Goals:

- To enable students to extend their programming ability by using advanced methods of representing data
- To solve interesting problems by using computer algorithms
- To gain exposure to college-level computer science material without needing to take an AP test (thus the rationale for AP weight).

Requirements:
Prerequisite: AP Computer Science A (recommended 80% or better).

Text:
Lewis, John, Chase, Joseph, and Sudol, Leigh Ann. Java Software Structures for AP* Computer Science AB, Pearson Education, Inc., c2005..

Key to Levels of Achievement (listed with each learning objective)

Awareness (A)	Students are introduced to concepts, forms, and patterns.
Learning (L)	Students are involved in a sequence of steps and practice activities which involve further development and allow for evaluation of process.
Understanding (U)	Students demonstrate ability to apply acquired concepts and skills to individual assignments and projects on an independent level.
Reinforcement (R)	Students maintain and broaden understanding of concepts and skills to accomplish tasks at a greater level of sophistication.

Unit	Num	Objective	Level	Content	Evaluation	Standard
I. Linked Lists and Pointers	1	Students will understand the nature of a linked list Abstract Data Type (ADT) conceptually.	L	● Representing linked structures ● Inserting and deleting while maintaining the original order ● Circular lists, doubly-linked lists, and header nodes	Homework assignments Unit Test	ISTE 3.d ISTE 4.b ISTE 4.d ISTE 6.c ISTE 6.d
	2	Students will know how to iterate over a **LinkedList** structure	L	● Instantiating iterators ● Using the **next** and **hasNext** methods	Unit Test	ISTE 3.d ISTE 4.b ISTE 4.d ISTE 6.c ISTE 6.d
	3	Students will use the ListNode class to implement functions and procedures to process linked lists	U	● Defining constructors ● Defining **add**, **addFirst**, **remove**, **removeLast**, **set**, and **get** methods ● Defining **contains**, **isEmpty**, and **size** methods ● Understanding the Big-O Notation for each method	Class presentations Unit Test	ISTE 3.d ISTE 4.b ISTE 4.d ISTE 6.c ISTE 6.d
	4	Students will use the **LinkedList** class to solve problems that are best done with sequential list structures such as merging two lists together.	R	● Algorithm design for doing operations on polynomials ● Algorithm design for doing a multiple choice tutorial	Programming projects - Polynomial Operations - Multiple Choice Tutorial Unit Test	ISTE 3.d ISTE 4.b ISTE 4.d ISTE 6.c ISTE 6.d
II. Stacks and Queues	5	Students will know the difference between using a Stack and a Queue conceptually	L	● Stack - LIFO Processing ● Queue - FIFO Processing ● Examples of stacks and queues in real-life	Unit Test	ISTE 3.d ISTE 4.b ISTE 4.d ISTE 6.c ISTE 6.d
	6	Students will implement a Stack class using static arrays and a **Queue** class using the **LinkedList** class	U	● Instantiating a Stack object ● Implementing the push, pop, peek, size, and **isEmpty** methods ● Instantiating a **Queue** object ● Implementing the **poll**, **offer**, **peek**, **isEmpty**, and **size**	Class presentations Unit Test	ISTE 3.d ISTE 4.b ISTE 4.d ISTE 6.c ISTE 6.d

				methods		
	7	Students will use a **Stack** class to solve application problems.	R	• Instantiating a **Stack** object • Using the **push**, **peek**, **pop**, **isEmpty**, and **size** methods	Programming project - Evaluating Expressions using Postfix Notation Unit Test	ISTE 3.d ISTE 4.b ISTE 4.d ISTE 6.c ISTE 6.d
	8	Students will use a **Queue** interface to solve application problems.	R	• Instantiating a **Queue** as a **LinkedList** object • Using the **offer**, **peek**, **poll**, **isEmpty**, and **size** methods	Programming project - Simulate waiting lines such as a timesharing system, an emergency room, a car wash, or a ticket counter Unit Test	ISTE 3.d ISTE 4.b ISTE 4.d ISTE 6.c ISTE 6.d
III. Binary and General Trees	9	Students will understand the nature of general and binary tree ADTs conceptually.	L	• Representing tree structures • Inserting and deleting while maintaining the original sorted order • Identifying characteristics of balanced, strictly binary, and complete tree structures	Homework assignments Unit Tesr	ISTE 3.d ISTE 4.b ISTE 4.d ISTE 6.c ISTE 6.d
	10	Students will know how to traverse over a Tree structure	L	• Doing preorder, inorder, and postorder traversals by hand • Creating recursive methods to do all three traversals of binary and general trees	Homework assignments Unit Test	ISTE 3.d ISTE 4.b ISTE 4.d ISTE 6.c ISTE 6.d
	11	Students will use the TreeNode class to implement functions and procedures to process binary trees	U	• Defining constructors • Defining **insert**, **delete**, **find**, **isEmpty**, and **size**~~, and is methods~~ • Understanding the Big-O Notation for each method	Class presentations Unit Test	ISTE 3.d ISTE 4.b ISTE 4.d ISTE 6.c ISTE 6.d

	12	Students will solve problems that are best done with binary tree structures.	R	• Algorithm design for doing binary search tree organization of data (e.g. student names) • Algorithm design for doing an intelligent data base	Programming projects - Dictionary of Java Identifiers - Student Names and Phone Numbers - 20 Questions Unit Test	ISTE 3.d ISTE 4.b ISTE 4.d ISTE 6.c ISTE 6.d
IV. Sets and Maps	13	Students will know the difference between using a **Set** and a **Map** conceptually	L	• Set – Unique objects only • Map – Unique keys that are associated with other objects • Examples of sets and maps in real-life	Unit Test	ISTE 3.d ISTE 4.b ISTE 4.d ISTE 6.c ISTE 6.d
	14	Students will know how to instantiate a set or a map using hash tables or binary trees	U	• Instantiating using **HashSet** or **HashMap** which maintains random order • Instantiating using **TreeSet** or **TreeMap** which uses sorted order	Unit Test	ISTE 3.d ISTE 4.b ISTE 4.d ISTE 6.c ISTE 6.d
	15	Students will write programs that use **Set** objects to solve problems.	R	• Instantiating a **Set** object • Using the methods **add**, **contains**, **isEmpty**, **remove**, **iterator**, and **size** • Algorithm design for programs	Programming projects - Cereal Box Prizes - Taxman Game - Word Wizard - Spanning Trees\ Unit Test	ISTE 3.d ISTE 4.b ISTE 4.d ISTE 6.c ISTE 6.d
	16	Students will write programs that use Maps to solve problems.	R	• Instantiating a Map object • Using the methods **get**, **containsKey**, **containsValue**, **keySet**, **remove**, **put**, **isEmpty**, and **size** • Algorithm design for programs	Programming projects - Tech Support - Address Book - Student Records Unit Test	ISTE 3.d ISTE 4.b ISTE 4.d ISTE 6.c ISTE 6.d