IMACS: AP Computer Science A

OVERVIEW
This course is a 34-week, 4 classroom hours per week course for students taking the College Board’s Advanced Placement Computer Science A exam.

It is an online course comprising narrative, 19 multiple choice tests, 204 exercises, 30 activities, and 8 labs. Many of the exercises and all the activities may be viewed as “mini-labs” requiring students to write and test programs or program segments. Students use a web browser to access a Java 1.7 web service in which the code they write is compiled and run on high-speed remote servers with results posted back to the browser. For the labs, students use an external IDE. Their progress is monitored in a real-time online grade book.

The course is divided into 11 units that are in turn organized into 4 sections.

RESOURCES AND SUPPLEMENTARY MATERIALS
- A recent internet browser, preferably Mozilla Firefox, Google Chrome, Opera, or Internet Explorer (version 9 or newer).
- The Eclipse or NetBeans Integrated Development Environment.

COURSE OUTLINE
Section 1: Introduction and Java Basics
This section introduces students to the course as a whole, and covers the essential hardware, software and system components of computer systems, and issues concerning the responsible use of computer technology. Basic elements of a Java program are covered, including variables, integer, string and array data types, arithmetic expressions, and relational and logical operators and expressions. In this section, code that is a necessary part of a Java program but which is unrelated to its specific function – including import statements, “public class MainClass”, “public static void main(String[] args)” and so on – is initially hidden from the students (but added automatically by the Java web service), so that students may focus on the basic language elements.

Unit 1: Introduction and Variables (Week 1)
Introduction; Computer Hardware, Software, Systems, Ethical and Responsible Use; Integers; Doubles; and Casting.

- Exercises 1 – 6.
- Quick Reference 1.
- Test 1.

Unit 2: Expressions (Weeks 2 – 4)
Arithmetic Expressions; Declaring and Assigning Values to Variables; Programming Shortcuts; Strings; Concatenation; String Methods; System.out Methods; Converting Between Numbers and Strings; Booleans; Relational Operators; Comparing Strings; Logical Operators; and Arrays.
• Exercises 7 – 50.
• Activities 1 – 5.
• Quick References 2 – 5.
• Tests 2 – 5.

**Unit 3: Program Control (Weeks 5 – 7)**
Conditional Statements; Blocks; Iteration; While Loops; For Loops; and For-each Loops.

• Exercises 51 – 74.
• Activities 6 – 10.
• Quick References 6 – 9.
• Tests 6 – 9.

**Unit 4: Methods (Weeks 8 – 11)**
Static Methods; Defining New Static Methods; The main Method; Java Comments; Multiple Variable Declarations; Overloaded Methods; and Recursive Methods.

• Exercises 75 – 92.
• Activities 11 – 13.
• Lab 1
• Quick References 10 and 11.
• Tests 10 and 11.

**Section 2: Object-Oriented Programming**
This section introduces students to object-oriented programming. The remaining “hidden” code that is being added automatically by the web service is revealed.

**Unit 5: Object-Oriented Programming Concepts (Weeks 12 – 16)**
An introduction to OOP in the context of Java; an overview of Classes and Instances; Simple Objects; A Person Class; A Point Class; Public Classes and the Java Compiler; The Java Compiler and the Virtual Machine; Errors, Exceptions, and Garbage Collection; Arrays and Objects; and ArrayLists.

• Exercises 93 – 115.
• Activities 14 – 16.
• Labs 2 and 3.
• Quick References 12 and 13
• Tests 12 and 13.

**Unit 6: Inheritance and Polymorphism (Weeks 17 – 19)**
Extending Classes; Class Hierarchies; Polymorphism; and Overriding Methods.

• Exercises 116 – 137.
• Activities 17 – 19.
• Lab 4.
• Quick References 14 and 15.
• Tests 14 and 15.
Unit 7: Class Definitions Revisited (Week 20)
Class Methods; Class Variables and Constants; final Block Variables; Multiple Constructors; Overloaded Instance Methods; Wrapper Classes; Access Modifiers; this; and Object Aliasing.

- Exercises 138 – 151.
- Activities 20 – 23.
- Quick Reference 16.
- Test 16.

Unit 8: Abstractions (Weeks 21 – 23)
Abstract Classes; Interfaces; and, in particular, the List<E> and Comparable<E> Interfaces.

- Exercises 152 – 166.
- Lab 5.
- Quick Reference 17.
- Test 17.

Section 3: Algorithms
This section introduces students to basic algorithms in the context of arrays, and includes a variety of standard searching and sorting algorithms.

Unit 9: Introduction to Algorithms (Week 24)
Algorithms; Traversals; Replacements; Insertions; and Deletions.

- Exercises 167 – 171.
- Activities 27 and 28.
- Lab 6.

Unit 10: Searching and Sorting, and Program Analysis (Weeks 25 – 29)
Sequential search; Binary search; Selection sort; Insertion sort; Merge sort; Assertions and Exceptions.

- Exercises 172 – 204.
- Activities 29 and 30.
- Labs 7 and 8.
- Quick References 18 and 19.
- Tests 18 and 19.

Section 4: Epilogue
The final section emphasizes the importance of review and practice in preparation for the Advanced Placement Exam.

Unit 11: Review and Practice (Week 30 – 34)
- Free-response Question Sets 1 – 3
- The eIMACS online edition of Be Prepared for the AP Computer Science Exam in Java, 6th ed., by Maria Litvin and Gary Litvin.
**CURRICULAR REQUIREMENTS**

**Requirement A:** The teacher has read the most recent *AP Computer Science A Course Description*.

The table below provides evidence that the designers of this online course have read the most recent *Course Description*. Many of the topics permeate several units of the course, and in such cases the Unit reference given here is either (a) the unit in which the corresponding term or concept is first defined or discussed or (b) the units(s) whose content best exemplifies the topic.

[In this table, the References column contains information that may be used to find sample online curriculum pages that are relevant to the topic. Index entries appear in the online index. Exercise, Activity, and Lab entries may be found using the corresponding items in the Finder submenu of the Options menu (select the Exercise, Activity, or Lab number from the flyout list and the relevant page will appear automatically).

<table>
<thead>
<tr>
<th>Topic</th>
<th>Unit</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>I. Object-Oriented Program Design</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>A. Program and Class Design</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. Problem analysis</td>
<td>[Unit 5]</td>
<td>Lab 2</td>
</tr>
<tr>
<td>2. Data abstraction and encapsulation.</td>
<td>[Unit 5]</td>
<td>Index: Encapsulation; Class; Data abstraction</td>
</tr>
<tr>
<td>3. Class specifications, interface specifications, relationships (“is-a,” “has-a”), and extension using inheritance.</td>
<td>[Units 5, 6, and 8]</td>
<td>Index: Class definition; Interface; Is-a relationship; Has-a relationship; extends; Class hierarchy; Exercise 116; Lab 4</td>
</tr>
<tr>
<td>5. Data representation and algorithms</td>
<td>[Units 5, 9 and 10]</td>
<td>Labs 2 and 3; Exercise 115</td>
</tr>
<tr>
<td>6. Functional decomposition</td>
<td>[Unit 7]</td>
<td>Index: Functional decomposition</td>
</tr>
<tr>
<td><strong>II. Program Implementation</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>A. Implementation techniques</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. Top-down</td>
<td>[Unit 5]</td>
<td>Index: Top-down programming</td>
</tr>
<tr>
<td>2. Bottom-up</td>
<td>[Unit 5]</td>
<td>Index: Bottom-up programming</td>
</tr>
<tr>
<td>3. Object-oriented</td>
<td>[Units 5-8]</td>
<td>Index: Object-oriented programming</td>
</tr>
<tr>
<td>4. Encapsulation and information hiding</td>
<td>[Unit 5]</td>
<td>Index: Encapsulation; Information hiding</td>
</tr>
<tr>
<td>5. Procedural abstraction</td>
<td>[Unit 4]</td>
<td>Index: Procedural abstraction</td>
</tr>
<tr>
<td><strong>B. Programming Constructs</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. Primitive types vs. reference types</td>
<td>[Unit 5]</td>
<td>Index: Data type, primitive; Data type, reference; Wrapper classes</td>
</tr>
</tbody>
</table>
2. Declaration
   a. Constants [Unit 7] Index: Class constant; final
   b. Variables [Unit 1, 5 and 7] Index: Variable; Class variable; Instance variable
   c. Methods and parameters [Unit 4] Index: Method, definition; Parameter list, formal
   d. Classes [Unit 5] Index: class
   e. Interfaces [Unit 8] Index: Interface

3. Text output using System.out.print and System.out.println [Unit 2] Index: System.out.print; System.out.println

4. Control
   a. Method call [Units 2, 4, and 7] Index: Method; Dot notation
   b. Sequential execution [Unit 3] Index: Sequential code execution
   d. Iteration [Unit 3] Index: Iteration
      Exercise 58
   e. Recursion [Unit 4] Index: Recursive method

C. Java library classes and interfaces included in the AP Java Subset
   
   Object (equals, toString) [Unit 6] Index: Object; equals [2]; toString [1]
   Integer (constructor, intValue, Integer.MIN_VALUE, Integer.MAX_VALUE) [Units 2, 5, and 7] Index: Integer; Integer.intValue; Integer.MIN_VALUE; Integer.MAX_VALUE
   Double (constructor, doubleValue) [Units 5 and 7] Index: Double; Double.doubleValue
   String (compareTo, length, substring [1- and 2-input], indexOf) [Unit 2] Index: String; compareTo [1]
   Math (abs [for int and for double], pow, sqrt, random) [Unit 4] Index: Math.abs; Math.pow; Math.sqrt; Math.random
   List<E> (size, add [1- and 2-input], get, set, remove) [Unit 8] Index: List<E>
   ArrayList<E> (size, add [1- and 2-input], get, set, remove) [Unit 5] Index: ArrayList<E> [1]

III. Program Analysis

A. Testing

1. Development of appropriate test cases, including boundary cases Many units Activity 1 Index: Boundary case
2. Unit testing [Unit 5] Lab 1 Index: Unit testing
3. Integration testing [Unit 10] Lab 3 Index: Integration testing
### B. Debugging

1. Error categories: compile-time, runtime, logic  
   [Unit 5]  
   Index: Errors, compile-time; Errors, logic; Exceptions, runtime

2. Error identification and correction  
   [Units 5 and 10]  
   Activity 29.3

3. Techniques such as using a debugger, adding extra output statements, or hand-tracing code  
   [Labs 1–8]  
   Lab 3

### C. Runtime exceptions

[Unit 10]  
Index: Exceptions; Exceptions, runtime [2]

### D. Program correctness

1. Pre- and post-conditions  
   [Unit 10]  
   Index: Precondition; Postcondition

2. Assertions  
   [Unit 10]  
   Index: Assertion

### E. Algorithm analysis

1. Statement execution counts  
   [Unit 10]  
   Exercise 178  
   Exercise 187  
   Index: Algorithm [3]

2. Informal running time comparisons  
   [Unit 10]  
   Lab 7  
   Index: Algorithm, order of growth of

### F. Numerical representations of integers

1. Representations of non-negative integers in different bases  
   [Unit 1]  
   Index: Number bases; Number representation

2. Implications of finite integer bounds  
   [Unit 1]  
   Index: INTEGER.MAX_VALUE

### IV. Standard Data Structures

A. Primitive data types (int, boolean, double)  
   [Unit 1]  
   Index: Data types, int; Data types, boolean; Data types, double

B. Strings  
   [Unit 2]  
   Index: Data types, String

C. Classes  
   [Unit 5]  
   Index: Class definition

D. Lists  
   [Unit 8]  
   Index: List<E>

E. Arrays (1-dimensional and 2-dimensional)  
   [Units 2 and 9]  
   Index: Array, one-dimensional; Array, two-dimensional  
   Lab 6

### V. Standard Operations and Algorithms

A. Operations on data structures

1. Traversals  
   [Unit 9]  
   Index: Traversing an array

2. Insertions  
   [Unit 9]  
   Index: Inserting an array element

3. Deletions  
   [Unit 9]  
   Index: Deleting an array element

B. Searching

1. Sequential  
   [Unit 10]  
   Index: Sequential search
2. Binary

C. Sorting

1. Selection
2. Insertion
3. Mergesort

VI. Computing in Context

A. System reliability
B. Privacy
C. Legal issues and intellectual property
D. Social and ethical ramifications of computer use

Requirement B: The course teaches students to design and implement computer-based solutions to problems.

Throughout this course, students are asked to design and implement programs each of which provides a computer-based solution to a problem. These problems occur in a wide variety of application areas. The following table provides illustrative examples:

<table>
<thead>
<tr>
<th>Application area/solution</th>
<th>Unit</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Biology/sampling</td>
<td>[Unit 5]</td>
<td>Exercise 115</td>
</tr>
<tr>
<td>Students develop a system for storing data collected by a group of biology researchers.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Poker</td>
<td>[Unit 5]</td>
<td>Lab 3</td>
</tr>
<tr>
<td>Students develop a solution for manipulating poker hands.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inventory tracking</td>
<td>[Unit 6]</td>
<td>Lab 4</td>
</tr>
<tr>
<td>Students develop a solution for tracking goods in a warehouse.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Turtle geometry</td>
<td>[Unit 8]</td>
<td>Lab 5</td>
</tr>
<tr>
<td>Students develop a solution for enabling a vector-graphics “turtle”.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Comparing sorts</td>
<td>[Unit 10]</td>
<td>Lab 7</td>
</tr>
<tr>
<td>Students incorporate various sorting algorithms into a testing environment.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Requirement C:** The course teaches students to use and implement commonly-used algorithms and data structures.

The following table lists some of the commonly-used algorithms and data structures covered by this course:

<table>
<thead>
<tr>
<th><strong>Topic</strong></th>
<th><strong>Unit</strong></th>
<th><strong>References</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Object-Oriented Program Design</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Problem analysis</td>
<td>[Unit 5]</td>
<td>Lab 2</td>
</tr>
<tr>
<td>Data abstraction and encapsulation</td>
<td>[Unit 5]</td>
<td>Index: <em>Class; Encapsulation; Data abstraction</em></td>
</tr>
<tr>
<td>Class specifications, interface specifications, relationships (&quot;is-a,&quot; &quot;has-a&quot;), and extension using inheritance</td>
<td>[Units 5, 6, and 8]</td>
<td>Index: <em>Class definition; Interface; Is-a relationship; Has-a relationship; Extends; Class hierarchy</em> Exercise 116 Lab 4</td>
</tr>
<tr>
<td>Code reuse</td>
<td>[Unit 4]</td>
<td>Index: <em>Procedural abstraction</em></td>
</tr>
<tr>
<td>Data representation and algorithms</td>
<td>[Units 5, 9, and 10]</td>
<td>Labs 2 and 3 Exercise 115</td>
</tr>
<tr>
<td>Functional decomposition</td>
<td>[Unit 7]</td>
<td>Index: <em>Functional decomposition</em></td>
</tr>
<tr>
<td><strong>Standard Algorithms</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Iteration</td>
<td>[Unit 3]</td>
<td>Index: <em>Iteration</em> Exercise 58</td>
</tr>
<tr>
<td>Traversals</td>
<td>[Unit 9]</td>
<td>Index: <em>Traversing an array</em></td>
</tr>
<tr>
<td>Replacements</td>
<td>[Unit 9]</td>
<td>Index: <em>Replacing an array element</em></td>
</tr>
<tr>
<td>Insertions</td>
<td>[Unit 9]</td>
<td>Index: <em>Inserting an array element</em></td>
</tr>
<tr>
<td>Deletions</td>
<td>[Unit 9]</td>
<td>Index: <em>Deleting an array element</em></td>
</tr>
<tr>
<td>Sequential search</td>
<td>[Unit 10]</td>
<td>Index: <em>Sequential search</em></td>
</tr>
<tr>
<td>Binary search</td>
<td>[Unit 10]</td>
<td>Index: <em>Binary search</em></td>
</tr>
<tr>
<td>Selection sort</td>
<td>[Unit 10]</td>
<td>Index: <em>Selection sort</em></td>
</tr>
<tr>
<td>Insertion sort</td>
<td>[Unit 10]</td>
<td>Index: <em>Insertion sort</em></td>
</tr>
<tr>
<td>Mergesort</td>
<td>[Unit 10]</td>
<td>Index: <em>Merge sort</em></td>
</tr>
<tr>
<td><strong>Data Structures</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Arrays (1-dimensional and 2-dimensional)</td>
<td>[Units 2 and 9]</td>
<td>Index: <em>Array, one-dimensional; Array, two-dimensional</em> Lab 6</td>
</tr>
<tr>
<td>ArrayLists</td>
<td>[Unit 5]</td>
<td>Index: <em>ArrayList&lt;E&gt;</em> Exercise 109 Lab 3</td>
</tr>
</tbody>
</table>
**Requirement D:** The course teaches students to develop and select appropriate algorithms and data structures to solve problems.

The development and selection of appropriate algorithms and data structures are common themes throughout the course. The table below lists some illustrative examples.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Unit</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Algorithms</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Recursive processes vs. iterative processes</td>
<td>[Units 4 and 9]</td>
<td>Exercises 91 and 92; Exercise 167; Index: Recursive method; Fibonacci number [2]</td>
</tr>
<tr>
<td>Comparison of sequential and binary search</td>
<td>[Unit 10]</td>
<td>Exercise 178; Index: Binary Search</td>
</tr>
<tr>
<td>Comparison of insertion and selection sort</td>
<td>[Unit 10]</td>
<td>Exercise 187; Exercise 192</td>
</tr>
<tr>
<td>Comparison of merge sort to insertion and selection sorting algorithms</td>
<td>[Unit 10]</td>
<td>Index: Merge sort, efficiency</td>
</tr>
<tr>
<td>Comparison of sorting algorithms</td>
<td>[Unit 10]</td>
<td>Lab 7; Index: Sorting algorithms, efficiency of</td>
</tr>
<tr>
<td><strong>Data Structures</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Arrays vs. ArrayLists</td>
<td>[Unit 5]</td>
<td>Index: Arrays and ArrayLists; Exercises 105 and 111; Lab 3</td>
</tr>
</tbody>
</table>

**Requirement E:** The course teaches students to code fluently in an object-oriented paradigm using the programming language Java.

The course teaches the Java programming language. See the following table.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Unit</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Object-Oriented Paradigm</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Object-oriented programming using Java</td>
<td>[Units 5-8]</td>
<td>Index: Object-oriented programming</td>
</tr>
<tr>
<td><strong>Object-Oriented Program Design</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Problem analysis</td>
<td>[Unit 5]</td>
<td>Lab 2</td>
</tr>
<tr>
<td>Data abstraction and encapsulation</td>
<td>[Unit 5]</td>
<td>Index: Class; Encapsulation; Data abstraction</td>
</tr>
<tr>
<td>Class specifications, interface specifications, relationships (&quot;is-a&quot;, &quot;has-a&quot;), and extension using inheritance</td>
<td>[Units 5, 6, and 8]</td>
<td>Index: Class definition; Interface; Is-a relationship; Has-a relationship; extends; Class hierarchy; Exercise 116; Lab 4</td>
</tr>
<tr>
<td>Code reuse</td>
<td>[Unit 4]</td>
<td>Index: Procedural abstraction</td>
</tr>
</tbody>
</table>
Requirement F: The course teaches students to use standard Java library classes from the AP Java Subset delineated in Appendix A of the AP Computer Science A Course Description.

The course provides a thorough treatment of all the standard Java library classes from the AP Java Subset. See the following table.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Unit</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java Library Classes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Object (equals, toString)</td>
<td>[Unit 6]</td>
<td>Index: Object; equals [2]; toString [1]</td>
</tr>
<tr>
<td>Integer (constructor, intValue, Integer.MIN_VALUE, Integer.MAX_VALUE)</td>
<td>[Unit 5]</td>
<td>Index: Integer; Integer.intValue; Integer.MIN_VALUE; Integer.MAX_VALUE</td>
</tr>
<tr>
<td>Double (constructor, doubleValue)</td>
<td>[Unit 5]</td>
<td>Index: Double; Double.doubleValue</td>
</tr>
<tr>
<td>String (compareTo, length, substring [1- and 2-input], indexOf)</td>
<td>[Unit 2]</td>
<td>Index: String; compareTo [1]</td>
</tr>
<tr>
<td>Math (abs [for int and for double], pow, sqrt, random)</td>
<td>[Unit 4]</td>
<td>Index: Math.abs; Math.pow; Math.sqrt; Math.random</td>
</tr>
<tr>
<td>List&lt;E&gt; (size, add [1- and 2-input], get, set, remove)</td>
<td>[Unit 8]</td>
<td>Index: List&lt;E&gt;</td>
</tr>
<tr>
<td>ArrayList&lt;E&gt; (size, add [1- and 2-input], get, set, remove)</td>
<td>[Unit 5]</td>
<td>Index: ArrayList&lt;E&gt; [1]</td>
</tr>
</tbody>
</table>

Requirement G: The course includes a structured lab component comprised of a minimum of 20 hours of hands-on lab experiences.

The course provides eight fully-elaborated labs, each with a number of optional extensions. Most students will require well over 20 hours to complete the required parts of these labs.

<table>
<thead>
<tr>
<th>Description</th>
<th>Unit</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chatter</td>
<td>[Unit 4]</td>
<td>Lab 1</td>
</tr>
<tr>
<td>Points and rectangles</td>
<td>[Unit 5]</td>
<td>Lab 2</td>
</tr>
<tr>
<td>Poker</td>
<td>[Unit 5]</td>
<td>Lab 3</td>
</tr>
</tbody>
</table>
poker hands, and that incorporate shuffling, dealing, and displaying hands in a graphics window.

**Inventory tracking**
Students develop classes that model the operation of a footwear warehouse.

[Unit 6] Lab 4

**Turtle geometry**
Students develop classes that provide a versatile vector graphics application in a GUI interface

[Unit 8] Lab 5

**Image processing**
Students develop an image processing application by extending a provided project by the addition of various methods that process 2-dimensional arrays.

[Unit 9] Lab 6

**Comparing sorts**
Students develop classes that implement a testing environment in which the various sorting algorithms may be compared in terms of their average running times.

[Unit 10] Lab 7

**Elevens**
Students develop classes that model playing cards and decks of cards in a different way than in Lab 3. By incorporating these into a provided project, they develop an application that plays various solitaire games.

[Unit 10] Lab 8

**Requirement H:** The course teaches students to recognize the ethical and social implications of computer use.

The following table lists the topics dealing with the responsible use of computers:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Unit</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>System reliability</td>
<td>[Unit 1]</td>
<td>Index: <em>Computing, system reliability</em></td>
</tr>
<tr>
<td>Privacy</td>
<td>[Unit 1]</td>
<td>Index: <em>Computing, privacy issues</em></td>
</tr>
<tr>
<td>Legal issues and intellectual property</td>
<td>[Unit 1]</td>
<td>Index: <em>Computing, legal issues</em></td>
</tr>
<tr>
<td>Social and ethical ramifications of computer use</td>
<td>[Unit 1]</td>
<td>Index: <em>Computer, social and ethical issues</em></td>
</tr>
</tbody>
</table>