Algorithms in C/C++
Focus on Computational Thinking Skill

Software: Microsoft Visual Studio Express
Model to follow:
1 No one moves onto the next Chapter until all assigned exercises are completed with excellent quality, ie. no ad-hoc wasted memory space, inefficient codes, often with optimization in mind.
2 This program focus on computational thinking and analysis, not just to learn the mechanics of the language.
3 That means all students have to complete the implementation of challenge presented in all levels, not just exploratory.
4 After mid-way of Level II, we do not go by the sequence of, but by requirement of given projects. For example, In Level II, students learn basics in recursive functions and must complete basic recursive functions. However, in Level III, they will return to Chapter 9 to cover quick sort algorithm and implement their own quick sort algorithm and compare performance as well.
5 Using debugger is always an essential component throughout all levels.
6 Through all exercises, students must perform their by divide and conquer method. This is for solving complex problems by breaking them down into simpler sub-problems, emphasizing reusability, portability, and scalability.

Annotation used in the syllabus listed in the next few pages.

Ch. == Chapters
(P) == Partial Chapter. However, topic covered is important at the time introduced, but simple enough to understand without reading the full chapter.
add’l == additional topics added in as topics is important and closely related to the topics taught at the time.
# Algorithms in C/C++ – Level I

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>C FUNDAMENTALS</td>
</tr>
<tr>
<td>2</td>
<td>Scope of Variables</td>
</tr>
<tr>
<td></td>
<td>Primitive data types, local vs global</td>
</tr>
<tr>
<td>3</td>
<td>FORMATTED INPUT/OUTPUT</td>
</tr>
<tr>
<td></td>
<td>including rudimentary in using #define preprocessor directive</td>
</tr>
<tr>
<td>4</td>
<td>EXPRESSIONS</td>
</tr>
<tr>
<td>5</td>
<td>Control structure with Selection (if/switch) statements</td>
</tr>
<tr>
<td>6</td>
<td>Loops Structure and Control structure</td>
</tr>
<tr>
<td>7</td>
<td>More in Data Types and their constraints.</td>
</tr>
<tr>
<td>8</td>
<td>Arrays (1-D and 2-D)</td>
</tr>
<tr>
<td>9</td>
<td>Program Design - multiple programs, include header file</td>
</tr>
<tr>
<td>10</td>
<td>Proper Development Habit - Coding Styles</td>
</tr>
<tr>
<td>11</td>
<td>Learn How to Write Robust and Readable Codes</td>
</tr>
<tr>
<td>12</td>
<td>Fundamentals in Debugger including break points, observation of variables, watch feature.</td>
</tr>
</tbody>
</table>

By the time this level is completed, students will have covered up to Chapter 8, and completed more than 40 programs and a few mini-projects such as:

**Sample Algorithms external to the book:**

- Prime number generator using Sieve of Eratosthenes Algorithm. At the end of the project, all programs must meet minimum performance requirement – 100,000 prime numbers within 3 msecs with our i5 processor CPU.
- Count of Sets of Anagrams through a series of multiple words.
- Write a Tic-tac-toe program to allow 2 humans to play against each other.
### Algorithms in C/C++ - Level II

Chapter 9, 10, 11, 12, 13, 16, 17, 22, 23

1. **Basic in Functions**
   - including pass by Reference vs. Value
   
2. **Pointers and Arrays of Pointers, etc.**

3. **Basics in Memory Allocation and dynamic storage allocation.**

4. **Strings Manipulations**

5. **ADTs - Structures, Unions, and Enumerations.**

6. **Sorting Algorithms:** (students need to implement the algorithm, not just use the intrinsic)
   - Insertion Sort Algorithm
   - Binary Search – (Tentative)
   - The Quicksort Algorithm

7. **Low Level Binary Operations**

8. **Program Design - multiple programs, external variables**
   - with proper usage of preprocessor directives

9. **Basics in File I/O**

10. **Random number generation & basics in performance evaluation.**

11. **including Command Line Arguments**

12. **Learning how to navigation around a shell system. (using ubuntu).** - (Tentative only)

13. **Writing a few common shell utility applications.**

   By the time this level is completed, students will have completed more than 20 programs with most focusing on indepth level of computational thinking and analytical skill to prepare them for Level III.

Sample Algorithms external to the book:

- Insert Sort.
- Work on a few problems sets from USACO/Bronze Level.
- Writing a utility function simulating unix util function such as grep, and wc

**** In order to participate in the USACO - Bronze, students must complete at least Level II.****
## Algorithms in C/C++ - III

Chapter 9, 17, 18, 19, 24, 26

1. Revising Iterative vs Recursion
   - Tail and head recursions
   - Write factorial/Fibonacci/sum of a series of natural numbers using recursion

2. Backtracking Algorithm - Depth First Search
   - MiniMax Search Algorithm - Tic-Tac-Toe
   - MiniMax with Alpha pruning Algorithm

3. FILE I/O (focus on analysis in efficiency in handle large files with simple memory management).

4. Pointers to Functions

5. Further discussion in Stack vs. Queue

6. Error and Signals Handling Interrupts

Other Possible projects in additional misc mini-projects provided:
   - Sieve Algorithm with Bits Segmentation method (optional)
   - Parse a postfix arithmetic formula (optional)
   - Tower of Hanoi (optional)

Note: Chapter 21, 23 to 27 are more like reference topics. By the time students complete Level III, they can easily browse through these topics on their own.

**** In order to participate in the USACO - Silver, students should continue to work on III and IV.
### Algorithms in C/C++ – IV

- **Advanced Algorithms Centric Level** *(outside the book)*

1. Implementation of Quicksort to handle all data types
   - Explore the power of void *
2. Pointers to Function
3. Program Design - Object Design Paradigm
   - Intro to Storage Classes and Inline functions
4. More on Stack ADT and Linked List
5. Backtracking Algorithm - Breath First Search – Shortest path
   - Dijkstra Algorithm
6. Hash Table Concepts
7. Minimum Spanning Tree Algorithm
8. Projects: Examples
   - Math Expression Parsing – parse the postfix
   - Maze project
   - Knapsack Problem
   - Huffman coding Encryption
   - N-Queen
   - A* algorithm

If this is implemented in robotics, this may be done in a maze setting with two possible challenge:

- **a)** Traverse the whole maze to seek for location of a certain target. Using BFS to go back to the start point to report the location of the target
- **b)** Have one robot to traverse a maze to seek for location of one target. Figure out the shortest path and send data remotely to home station robot. This home station robot will send traverse based on the path given by the first robot.

Note that these may be done in a different order depending on whether they will use the BFS in the RCJ- Maze Project.

**** In order to participate in the USACO - Gold, students should complete IV.

This level will cover College Level Computer Science Data Structure Class criteria and beyond.