# Table of Content

**Preface** .................................................................................................................. P.1

**Characteristics** .. 2

**Requirement for Levels promotion** .. 3

**Honor Code** .. 3

**Books** .. 4

**Software** .. 4

**Eligibility in Participating Competitions** .. 4

**Why C** .. 5

**Level B**.................................................................................................................. P.6

**Level I**.................................................................................................................. P.8

**Level II**

**II- C** ................................................................................................................. P.11

**II- C++** .............................................................................................................. P.14

**II- Java** .............................................................................................................. P.16

**Level III**............................................................................................................ P.17

**Level IV**............................................................................................................ P.19

Last update: September 1st, 2017
**PREFACE**

Computer Science skill should go beyond just programming itself but much more importantly in problems solving skill with computational thinking even for grade schools.

Automation is entrenched in our daily lives in the era of digital age. Computer Science with computational thinking is indispensable for strengthening the foundation. This sub-group aims to build this core foundation and sharpen students' problem solving skills in this digital world, no matter whether in engineering, or even liberal arts area.

Storming Robots utilizes Robotics to animate problem solving effort starting from Grade 4. However, we encourage students to study in this Algorithms in C/C++ Track starting from Grade 8.

This document lists the Syllabus covered in this Track. This syllabus consists of four levels targeting for students from Grade 8+. Levels with college level computer science topics —data structure, introduction to algorithms with combinatorial optimization, and complexity analysis.

**CHARACTERISTICS**

1) Focus on problems solving/software development skill, so students will not work with the physical robot.
2) All levels stress in Computational Thinking and Efficiency.
3) Progress is self-paced with excellent work quality. Exercises are often adjusted for varying levels of students achievement.
4) All assigned exercises should be completed with excellent quality, with efficiency, memory consumption, readability, maintainability, etc.
5) Students will be allowed to move quickly to the next concept only after
they demonstrate satisfactory level of understanding through their homework and pop-quiz in class. Those with gaps in their prerequisite knowledge will receive additional exercises to address the shortcomings.

6) Allow students to embed other competitions in-between levels.

**Requirement for Level Promotion:**

1) This track is broken down into four levels. Students must demonstrate they can complete all givens exercises and make all corrections required.

2) Pop-quiz problems may be assigned to students to complete in class in order to prove their understanding.

Students will be encouraged to participate the USA Computing Olympiad online exams (far more demanding than AP CS) as performance gauges; although this is not mandatory from SR. Levels in USACO also help us to decide which competitions/exams/advanced projects students may participate.

**Honor Code**

Programming assignments are pledged work and are bound by the honor code. To simply put, the violation may be in any of the following forms:

- Claim someone else's work as their own.
- Edit someone else’s work by simply changing variables or style, etc., and claim to be the result of your own work.
- Complete assigned work with so much substantial help from others that you cannot even explain your own work; but still claim it is the result of your own work.

If found honor code is violated, parents will be informed. Students will not be allowed to participate in any competitions with Storming Robots.
**BOOKS FOR:**

- **Level II-OOPS—C++**:

**SOFTWARE:**

- (Windows OS) Eclipse or Microsoft Visual Studio Community Version.
- (Apple OS) Eclipse, Or XCode. Please do note that we shall not be able to offer installation advise if you use Apple OS.

**ELIGIBILITY IN PARTICIPATING COMPETITIONS**

<table>
<thead>
<tr>
<th>SP = Somewhat proficient</th>
<th>HP = Highly Proficient</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Minimum Required Levels</strong></td>
<td>B</td>
</tr>
<tr>
<td>USACO Silver</td>
<td>—</td>
</tr>
<tr>
<td>USACO Gold</td>
<td>—</td>
</tr>
<tr>
<td>USACO Platinum 3</td>
<td>—</td>
</tr>
<tr>
<td>RCJ Line 1</td>
<td>SP</td>
</tr>
<tr>
<td>RCJ Maze</td>
<td>—</td>
</tr>
<tr>
<td>RCJ Soccer</td>
<td>—</td>
</tr>
<tr>
<td>ZeroRobotics 2</td>
<td>—</td>
</tr>
</tbody>
</table>

1. Although it states minimum requirement is to have “somewhat proficient”, higher level is a must in order to do well in the competition.
2. Require minimum Pre-Cal & Algebra II. Linear Algebra will be helpful.
3. USACO is designed to be challenging even for the best students. There are only a very small handful of students in USA obtaining Platinum.

To obtain more details about each competition: [http://compete.stormingrobots.com](http://compete.stormingrobots.com).
WHY C?

C programming allows high flexibility and the power to perform closer to the machine than “modern” languages. It is still often used in embedded systems or high-performance computing. It is considered to be lingua franca of programming. We focus on educating students to “understand” the underlying structure, think efficiency and analysis with algorithms; not to learn a specific language. This will equip students the skill to adapt the mechanics of new modern language and exercise with various algorithms and optimization techniques.

The following survey was done by IEEE:

The Top Programming Languages 2017 Survey by IEEE.

Rank by Embedded Systems realm:

<table>
<thead>
<tr>
<th>Language Rank</th>
<th>Types</th>
<th>Spectrum Ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. C</td>
<td>![Phone, Desktop]</td>
<td>99.7</td>
</tr>
<tr>
<td>2. C++</td>
<td>![Phone, Desktop]</td>
<td>97.2</td>
</tr>
<tr>
<td>3. Arduino</td>
<td>![Phone]</td>
<td>73.0</td>
</tr>
<tr>
<td>4. Assembly</td>
<td>![Phone]</td>
<td>72.1</td>
</tr>
<tr>
<td>5. Haskell</td>
<td>![Desktop]</td>
<td>48.5</td>
</tr>
</tbody>
</table>

Note: Arduino is a platform having its own set of APIs, not a programming language. The IEEE article included it because Arduino has become the basis of a large volume of prototype devices.

Rank by Overall:

<table>
<thead>
<tr>
<th>Language Rank</th>
<th>Types</th>
<th>Spectrum Ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Python</td>
<td>![Globe, Desktop]</td>
<td>100.0</td>
</tr>
<tr>
<td>2. C</td>
<td>![Phone, Desktop]</td>
<td>99.7</td>
</tr>
<tr>
<td>3. Java</td>
<td>![Globe, Phone]</td>
<td>99.4</td>
</tr>
<tr>
<td>4. C++</td>
<td>![Phone, Desktop]</td>
<td>97.2</td>
</tr>
<tr>
<td>5. C#</td>
<td>![Globe, Phone]</td>
<td>88.6</td>
</tr>
</tbody>
</table>
This covers the bare fundamentals of computer programming from basic expressions to compound control structure.

**Learning Outcome**

∞ By the time this level is completed, students should have completed 20+/- programs up to Chapter 6 from the K.N. King book. This is not a hard-set number because more exercises will be given if necessary in order to strengthen one’s understanding in a certain subject matter; or vice versa.

∞ Possess the knowledge in the fundamentals of programming with an emphasis on producing clear, robust, and reasonably efficient code using top-down design, informal analysis, and effective testing and debugging.

∞ Proper Mindset in computational thinking and analysis.

∞ Know how to use Debugger for trouble shooting.

∞ At completion of Level B, students should be able to master simple control structure, take a straightforward problem set to resolve it with computer programming in C.

If you join this class with prior programming background in RobotC, it will help to accelerate at the first four chapters.

**What will be covered**

1. Introducing C and Strengths and Weaknesses of C - Chapter 1
2. How to use the Microsoft Visual C/C++ Compiler IDE.
3. C Fundamentals in writing Simple Program - Chapter 2
4. Standard Formatted Input/Output - Chapter 3
5. Expressions (simple to compound) - Chapter 4
6. Rudimentary level in using functions() – class note
7. Selection/ if - Control Statements | Boolean Expressions - Chapter 5
8. Introduction of State Tables/Diagrams. Simple idea in State machine—more example exercises using “switch” to implement state tables. More in depth will be covered in Later Levels.
9. Loops Control Structure - Chapter 6
10. Further understanding in data types, signed vs. unsigned, further reinforcement in Loops Control Structure - Chapter 7

*** Reminder: Instructor will check on student’s every single program. Completion means satisfactory approved by the instructors. For example, codes cannot contain redundancy and high inefficiency. In addition, when instructor finds students still having difficulty in completing problems without substantial help, students will be given additional exercises for reinforcement.
All students who complete this level should participate the USACO online exams. USACO contests are designed to challenge even the very best students. For the students who have the foundation will still need to take a good deal of practice to excel at them. Completion of Level I is required for all students who wish to participate in any Open level of Robotics Competition.

**Learning Outcome:**

- Should be able to tackle Silver Level in the USACO. Based on our history, all students who demonstrate high proficiency in achieving Level I and have done some practices have promoted to at least Silver Level.

- You not only should have covered the core control structure foundation required in AP Computer Science, but also will be stronger in programming analysis. Most programs exercises you have completed in this level exercise higher analytical skill required in AP Comp. Sci. The only additional content will be in the basics in OOPS design pattern. That will be covered in Level II—OOPS.

- Students demonstrated high proficiency in this level will be able to self-study for AP Computer Science—A with ease.

- Continue to improve in the habitual Mindset in computational thinking and analysis.

- Become more efficiency in Debugger including break points, observation of variables, watch feature.

- By the time this level is completed, students should have completed 20+ programs. This is not a hard-set number because more exercises will be given if necessary in order to strengthen one’s understanding in a certain subject matter.

- Build fundamentals of programming with an emphasis on producing clear, robust, and reasonably efficient code using top-down design, informal analysis, and effective testing and debugging. This will help them immensely in their future endeavor in any engineering programs.
Should expect to work additional exercises outside the book as well.
Further reinforce algorithmic problems solving skills with computer programming.

**WHAT WILL BE COVERED**

1) **Arrays : One and Multi- Dimensional** - Chapter 8
2) **More in creating Functions** - Chapter 9 (partial)
3) **Touch on the process from compilation to linkage.**
4) **Basics in Creating a project with multiple programs** - Chapter 10
   - Simple Project creation with multiple programs, such as header files, basic directives, extern, function prototypes, etc.
5) **Basics concepts in memory pointers with 1D and 2D array** – Chapter 11 & 12
   - Review Exercises to summarize Usage of array and pointers. Exercises external to the textbook.
6) **String manipulation** – Chapter 13
   - using C string vs memory library, Arrays of Strings, Dynamic allocation. Exercises external to the textbook.
7) **Recursion**—Exercises external to the textbook.
8) **Bitwise operations**—Chapter 20
   - Using bits in structure
   - Writing basic arithmetic calculation with using bits operation only.
9) **Basics in using Structures, Unions, and Enumerations**—chapter 16.
10) **Basics in Sorting:**
    - Write your own insertion sort.
    - Learn about Quick Sorting Algorithm, and use qsort().
11) **Basic File I/O** — Chapter 22
    - With C++ standard I/O (external to textbook)
12) **Simple Command-Line Operations** –
    - Compile and link programs with the IDE
    - ow to use system path, environment variables, etc.

**TO CONCLUDE THIS LEVEL:**

*** Mastery of all givens exercises from Ch. 8, 9 (partial), 10, 11, 12, 15, 16 and given exercises external to the textbook will conclude Level I. The external exercises include:

   A) Array
   B) String
   C) Pointers
   D) Structure
   E) Recursion

*** Reminder: Instructor will check on student’s every single program. Completion means satisfactory approved by the instructors. For example, codes cannot contain redundancy. In addition, when instructor finds students still having difficulty in completing problems, students will be given additional exercises.

*** May practice a few USACO Bronze and/or Silver level problems to prepare for USACO, depending on the timeframe.

*** At this level, all should also meet the prescribed Programming Style

There will be pop quiz given at random time to students to ensure satisfactory proficiency in the covered concepts. If students fail to perform well for the pop quiz, he/she will not be allowed to promote to the next level.
LEARNING OUTCOME:

∞ By the time this level is completed, students should have completed 20+ programs. You should expect each exercise will be longer than those exercises in Level I.

∞ Understand Fundamentals in College level Data Structure.

∞ You should be ready to practice exercises posted on www.usaco.org—Bronze and Silver level. Students should start doing practice themselves as there are many exercises available online at www.usaco.org.

∞ During November to April, students are encouraged to continue working on USACO problems set, as well as take the Bronze to Silver Level online Exam.

WHAT WILL BE COVERED

1) Writing MiniMax and Alpha Beta Pruning. (Class Note)
2) Preprocessor - Chapter 14—Conditional Compilation, such as #if, #elif, #line, etc. Its role in compilation process.
3) Advanced Uses of Pointers — Chapter 17
   a. Dynamic memory allocation
   b. Pointer to function
   c. Pointer to pointer
   d. Generic programming with void* and pointer to function—Complete Qsort using void *.
3) Storage Classes— Chapter 18
   Auto | static | extern | register | inline
4) Abstract Data Type – Chapter 19
   a. Linked List— single | double | circular
   b. Basic Stack (LIFO)— Push / Pop
c. Queue (FIFO) concepts— Enqueue / Dequeue (with/without own memory pool)
e. Explore Signals Handling – Chapter 19
f. Optional: More on low level knowledge (only for those who will work with micro-controller):
   ∞ Volatile memory
   ∞ Using offsetof() (additional class note)

9) Explore Deterministic Finite State Machine | Flowchart (additional class note)

**ADDITIONAL PROJECTS:**

∞ Practice on several USACO Silver and/or Gold level exercises.

This level will conclude the usage of the Dr. King’s C Book. Students should skim thru Ch. 21, 23 to 25-27 on their own, as they all should be used for reference and self-study purpose only.

You will need to a couple of new books in the future levels.

**Important Note before the next step:**

The immediate next level is II-OOPS with either C++ or Java. Before you proceed, you should read thru initial description before the next step. The next step can be one of the following:

1) Continue OOPS with C++; or
2) Continue OOPS with Java; or
3) Go right to Level III
This level does not necessarily mean being more complex topics than Level II. You may actually go right to Level III without taking Level II—OOPS. Level II—OOPS stands for Object-Oriented Programming and System. It will mainly focus on Object-Oriented Programming Paradigm.

Level II will be covered in either C++ or Java. This depends on each individual’s choice. With strong Level II foundation, students should go thru this section rather short time. It does not matter it will be done in C++ or Java.

**LEARNING OUTCOME:**

By the time this level is completed, students should have a good basic grasp of Object-Oriented Design Pattern. If you are going into computer engineering or any courseware which involves working with microcontroller libraries, completion of this will help you greatly in working with these libraries. (See: **Why C** section)

**Choosing II-OOPS-C++**

For students who are gearing towards Robotics Engineering in the future, you should use C++ as most of libraries written for embedded systems are also written in C++, not just C.

**II-OOPS-Java**

*If you aim to learn AP CS (grade 9+) with SR, you may want to consider Java is the recommended language. However, most of our Level II students will simply self-study Java instead.*
**Books:**

- If you choose Java, you should purchase:
  - Online: https://docs.oracle.com/javase/tutorial/java/index.html
  - Class note.

- If you choose C++, you should purchase:

**Topics to be Covered in OOPS**

1. Basic Class/Object Programming Structure
2. Basic Instinct Wrapper Classes
3. Constructors, Destructors
4. Overloading
5. Strings Manipulation
6. Basic File I/O Manipulation
7. Object Oriented Pattern Principles
   - Encapsulation
   - Data Hiding
   - Inheritance
   - Polymorphism
7. Recursion
8. Basics in Search and Sorting Algorithms
   a. Binary Search
   b. Sorting: Selection Sort | Insertion Sort | Quick Sort | Merge Sort
8. Building a project with multiple files
More Details on Sequence if you choose C++:

**Standard Library Abstractions**

1) Basic C++ standard I/O, very fundamentals mechanics—Ch. 0
2) Strings vs Arrays—Ch. 1 to 2
3) Vectors —Ch. 3
4) Reference Pointers Type, and Exception Handling—Ch. 4
5) Sequential containers (Iterators) and strings—Ch. 5
6) Some commonly used methods in Algorithm library—Ch. 6
7) Performance on using map< >—Ch. 7

*** Note: The optional reference book “C++ Primer” is a very helpful resource organized in proper sequence. At this point, I assume that you have become a quite efficient self-learner. Depending on your goal before college, projects, etc., we may skip Ch. 8 to 15 and jump to Level III. Therefore, the amount of coverage on the remaining chapters will depend on the individual student’s strength and interest, and availability outside school work.

**Defining Your Own Library Abstractions**

1) Generic Programming —Ch. 8
2) Creating New Class, Data hiding, Types—Ch. 9
3) Memory Management and low-level data structures—Ch. 10
4) More on ADTs —Ch. 11
5) Overload and type conversions —Ch. 12
6) Inheritance, Polymorphism, and Dynamic Binding —Ch. 13
7) More advanced topics on Data Cloning —Ch. 14
8) Virtual | Friend Types —Ch. 15
More Details on Sequence if you choose Java:

Working exercises are from class notes and SR’s online document. Use class notes, and Online Official Java Document from docs.oracle.com as frequent reference.

Following list the summary of all topics and links to Oracle’s site for your reference.

<table>
<thead>
<tr>
<th>Classes and Objects</th>
<th>Interfaces and Inheritance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classes</td>
<td>Interfaces</td>
</tr>
<tr>
<td>Declaring Classes</td>
<td>Defining an Interface</td>
</tr>
<tr>
<td>Declaring Member Variables</td>
<td>Implementing an Interface</td>
</tr>
<tr>
<td>Defining Methods</td>
<td>Using an Interface as a Type</td>
</tr>
<tr>
<td>Providing Constructors for Your Classes</td>
<td>Evolving Interfaces</td>
</tr>
<tr>
<td>Passing Information to a Method or a Constructor</td>
<td>Default Methods</td>
</tr>
<tr>
<td>Objects</td>
<td>Summary of Interfaces</td>
</tr>
<tr>
<td>Creating Objects</td>
<td>Inheritance</td>
</tr>
<tr>
<td>Using Objects</td>
<td>Multiple Inheritance of State, Implementation, and Type</td>
</tr>
<tr>
<td>More on Classes</td>
<td>Overriding and Hiding Methods</td>
</tr>
<tr>
<td>Returning a Value from a Method</td>
<td>Polymorphism</td>
</tr>
<tr>
<td>Using the this Keyword</td>
<td>Hiding Fields</td>
</tr>
<tr>
<td>Controlling Access to Members of a Class</td>
<td>Using the Keyword super</td>
</tr>
<tr>
<td></td>
<td>Object as a Superclass</td>
</tr>
<tr>
<td>Summary of Creating and Using Classes and Objects</td>
<td>Writing Final Classes and Methods</td>
</tr>
<tr>
<td></td>
<td>Abstract Methods and Classes</td>
</tr>
<tr>
<td></td>
<td>Summary of Inheritance</td>
</tr>
</tbody>
</table>

Numbers and Strings | (Note: This covers beyond AP CS. This will fulfill the requirement for Java Programming Level I Exam Certification by Oracle.) |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Numbers</td>
<td>Link to Additional Exercises. )</td>
</tr>
<tr>
<td>The Numbers Classes</td>
<td>Sorting: Sequential and Binary Search</td>
</tr>
<tr>
<td>Formatting Numeric Print Output</td>
<td>Sorting: Selection</td>
</tr>
<tr>
<td>Beyond Basic Arithmetic</td>
<td></td>
</tr>
<tr>
<td>Summary of Numbers</td>
<td></td>
</tr>
<tr>
<td>Questions and Exercises</td>
<td></td>
</tr>
<tr>
<td>Characters</td>
<td></td>
</tr>
<tr>
<td>Strings</td>
<td></td>
</tr>
<tr>
<td>Converting Between Numbers and Strings</td>
<td></td>
</tr>
<tr>
<td>Manipulating Characters in a String</td>
<td></td>
</tr>
<tr>
<td>Comparing Strings and Portions of Strings</td>
<td></td>
</tr>
</tbody>
</table>
We shall take a more pragmatic approach, ie. often target to an application. This will help to keep students more engaged and easier to understand the purpose and use of common programming techniques.

Students should expect projects in this level is FAR more complex than Level II. Level III+ work will put you a substantial head start in college if you will major in Computer Science or even Computer Engineering.

**Do note that Level II—(C) is a pre-requisite of Level III, II-OOPS is not required before Level III. However, C++ is a superset of C, it means it will contain many important features which will reduce your work load vs using just C, especially in arrays and memory management.**

**LEARNING OUTCOME:**

Students will have covered the essential data structures and most commonly discussed algorithms today. Students who are able to complete this level will have proven themselves excellence in programming levels like a software mini-professional. All of the students who can reach this level should gain a highly competitive edge in college internship programs. Many of our alumni reported to us that they got internship interview even at College Freshmen year.

If students possess excellent “Analytical” math level, they should do well in obtaining Gold level for [www.usaco.org](http://www.usaco.org). However, nothing can replace: Practice! Practice! Practice!

**Books:**

- This book will be used for Level III & IV with additional notes.
What will be covered

1) Introduction to Data Structures and Algorithms — Ch. 1
2) Advanced Pointer Manipulation — Ch. 2
3) Dynamic Programming Techniques — Class Notes
4) KnapSack Problem
5) Recursive Binary Tree Algorithms — Class Notes
6) Analysis of Algorithms — Ch. 4
   - Running time / Worst-case analysis
   - Big-O notation
   - Computational Complexity
   - Rate of Growth
   - Asymptotic Analysis
   - Analysis of Binary Search, vs QuickSort vs MergeSort (class note)
7) Data Structures
   - More in-depth in Linked List — Ch. 5
   - More in-depth in Stacks and Queues — Ch. 6
8) Basic Backtracking Algorithms
   - Depth First Search — MinMax Algorithm with Tic-Tac-Toe with and without alpha-beta pruning.
   - Breath First Search — This may be done in a maze setting with two possible challenge: Traverse the whole maze to seek for location of a certain target. BFS to go back to the start point to report the location of the targets.
9) More on Data Structures
   - Hash Tables — Ch. 8
   - Trees — Ch. 9
   - Heaps and Priority Queue — Ch. 10

Optional exercises:

- Backtracking algorithm with NQueen Algorithm
- Tower of Hanoi
- Math Expression Parsing — postfix notation
LEARNING OUTCOME:
At this level, students are readily to tackle USACO—Gold or even Platinum level if you possess excellent analysis in Mathematics as well. I am not referring to math like Calculus, nor linear algebra, but “Analysis” skill in combinatorial math.

WHAT WILL BE COVERED:
1) Numerical Methods—Ch. 13
2) Data Compression—Ch. 14
   – Huffman Coding Algorithm
   – LZ77 (basis for LZW)
3) Data Encryption — Ch. 15
   – DES vs RSA
4) String Algorithms—Class Notes
   – Longest Increasing Sequence
   – Longest Common Sequence
5) More on Data Structure :
   – Sets—Ch. 7
   – Graphs—Ch. 11
6) Graph Algorithms—Ch.16
   – Minimum Spanning Trees—Kruskal
   – Minimum Spanning Trees—Prim
   – Dijkstra—Shortest Path
   – Travel Salesman Problem
7) Geometric Algorithms—Ch. 17
   – Convex Hulls
9) Travelling Salesman Problem.
10) Job Scheduling Problem

Optional:
∞ Explore Parallel programming with semaphore.
∞ A* algorithm (Path finding)

SUMMARY OF MAJOR ALGORITHMS COVERED IN LEVEL III & IV

Greedy / Graph / Tree Algorithms:
1) Huffman Coding Compression
2) LIS and LCS
3) Prim's Minimal Spanning Tree
4) Kruskal’s Minimal Spanning Tree
5) Dijkstra’s Shortest path
6) Travelling Salesman Problem
7) Job Scheduling Problem

Divide and Conquer:
1) Merge Sort
2) Quick Sort
3) Binary Search

Combinatorial optimization
1) Fibonacci number series
2) Knapsack problem

Pattern Recognition
1) Nearest—Neighbour
2) K-NN

Others (Tentative):
1) Matrix Multiplication
2) Math Expression Parsing (postfix)
3) MD5
4) All pair shortest path by Floyd-Warshall
5) A* algorithm