Subject Description Form

Subject Code	COMP 1D04
Subject Title	From Scratch to Apps: Foundations of Computational Thinking and Literacy for Problem Solving
Credit Value	3
Level	1
Medium of Instruction	English
Pre-requisite and/or Exclusion(s)	Pre-requisites: Nil Exclusion: COMP 1901 Seminars and Topics in Information Technology COMP 1011 Programming Fundamentals
Objectives	Specific objectives of the subject: This subject combines relevant content from two subjects for the Broad Discipline of Computing – the Freshman Seminar and Programming Fundamentals subjects – into a single subject that is intended specifically for non-Computing students. The objectives are (1) to introduce non-major students to core concepts of computing as problem solving principles, (2) to give them a broad-based foundation in computational thinking and logical literacy, and (3) to relate the concepts to real-life problems in a broad range of domains. Students are not expected to have any background in Computing or programming. The subject will cover computational concepts, including sequential logic, abstraction and problem solving techniques through a hands-on, example-based learning approach. The hands-on activities will start with a graphical-based, drag-and-drop programming language such as Scratch, Alice, App Inventor and/or programming envirnoments such as Excel macros. A gradual, guided transition will then be made to high-level programming language such as Python or php. Examples and projects will be centered around applying the computational thinking skills acquired in class to problems inspired from real-world domains, such as music, movies, social networks, finance, etc.
Intended Learning Outcomes	Upon completion of the subject, students will be able to: a. Understand the basic concepts of computational thinking, including sequential logic, abstractions, conceptualization and problem-solving; b. Possess the ability to design and develop programs to solve basic computational problems; c. Possess the ability to model real-life problems as computational problems; and d. Possess the ability to extend their knowledge towards learning other high level programming languages independently. Relationship between the learning outcomes with the following three essential features: Literacy, Higher order thinking, and Life-long learning Literacy: Learning outcomes (c) and (d) will require students to possess and develop basic English literacy. Students will be also assigned pre-lecture readings from popular computing magazines and journals to broaden their knowledge and exposure to the

<table>
<tr><td></td><td>field.

In addition, learning outcomes (a)-(c) are designed to teach and to train students' computational and logical literacy. Substantive practice will be required from students through reading, designing and implementing programs.

Higher-Order Thinking:
Learning outcomes (b) and (c) are designed to teach and to train students' higher-order thinking and problem-solving skills by requiring them to apply basic computational thinking methods and concepts to solving problems. These problems will range from straightforward computational problems to more complex problems encountered in daily life.

Life-Long Learning:
Learning outcomes (c) and (d) are designed to train students and provide them with the foundations for independent life-long learning. Through demonstrating the link between computational thinking and real-world problems, the subject aims to motivate students to pursue continuous learning by making relevant the subject matter to daily life. Through demonstrating how the foundations of the computational concepts extend to all programming languages, the subject also motivates and prepares students to learn new computer programming languages and concepts for their needs in future. In addition, to broaden students' knowledge and exposure to the field, pre-lecture readings from popular computing magazines and journals, such as Communications of the ACM, IEEE Computer, IEEE Spectrum, etc will be assigned.</td></tr>
<tr><td>Subject Synopsis/ Indicative Syllabus</td><td>1. Overview of computational thinking and problem solving. Application of sequential logic to computational decomposition of problems.

2. Abstractions of real-life problems as basic programming concepts. Conditions, selection controls and looping, arrays and functions. Representations of real-life entities as Abstractions and Data.

3. Problem solving procedures and tools. Simple algorithms, problem decomposition, solution design. Implementation, testing and debugging.

4. Application of computational techniques to various domains, including knowledge management, education, entertainment, digital edutainment, manufacturing, geo-informatics, bio-informatics, etc.</td></tr>
<tr><td>Teaching/Learning Methodology</td><td>The course material will be delivered mainly in a hands-on, example-based format. A combination of lectures and workshops will be used to provide students with the requisite knowledge and reinforcing absorption and retention of concepts through immediate practice and use.</td></tr>
<tr><td>Assessment Methods in Alignment with Intended Learning Outcomes</td><td>
<table>
<tr><td rowspan="2">Specific assessment methods/tasks</td><td rowspan="2">% weighting</td><td colspan="4">Intended subject learning outcomes to be assessed (Please tick as appropriate)</td></tr>
<tr><td>a</td><td>b</td><td>c</td><td>d</td></tr>
<tr><td>1. Assignments</td><td>50</td><td>X</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>2. Tests/quizzes</td><td>50</td><td>X</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>Total</td><td>100 %</td><td colspan="4"></td></tr>
</table>

The assessment for this subject will include tests/quizzes and assignments. Both</td></tr>
</table>

<table>
<tr><td></td><td colspan="2">components are designed to evaluate students' grasp of all the learning outcomes, including their grasp of basic computational thinking concepts and problem-solving.

To reinforce students' learning and retention of the subject content, the course will include around 6-8 short quizzes over the semester. Grade guarantees will be attached to certain quizzes to allow all students a chance to get a good grade in the subject, even those who do not have prior computing or programming experience, or who start off the subject slowly.

Some assignments will be group assignments to allow students to exercise their problem-solving skills in a more in-depth and creative manner.</td></tr>
<tr><td rowspan="7">Student Study Effort Expected</td><td>Class contact:</td><td></td></tr>
<tr><td>▪ Lectures, Workshops and Labs</td><td>26 Hrs.</td></tr>
<tr><td>▪ Tutorials</td><td>13 Hrs.</td></tr>
<tr><td>Other student study effort:</td><td></td></tr>
<tr><td>▪ Self study</td><td>31 Hrs.</td></tr>
<tr><td>▪ Assignments, exercises and projects</td><td>35 Hrs.</td></tr>
<tr><td>Total student study effort</td><td>105 Hrs.</td></tr>
<tr><td>Reading List and Reference</td><td colspan="2">1. R. Kowalski, Computational Logic and Human Thinking: How to be Artificially Intelligent Cambridge University Press; first edition (August 22, 2011).
2. M. Badger, Scratch 1.4: A Beginner's Guide. Packt Publishing (July 17, 2009).
3. T. Gaddis, Starting Out with Alice: A Visual Introduction to Programming. Addison-Wesley, 2nd Edition(2010)
4. J. Zelle, Python Programming: An Introduction to Computer Science, Franklin, Beedle & Associates, Second edition (May 18, 2010)
5. S. Welch, From Idea to App: Creating iOS UI, animations, and gestures (Voices That Matter), New Riders Press (2011)
6. Appropriate articles from Communications of the ACM, IEEE Computer and IEEE Spectrum. (Approximately 1 article per 1-2 lectures).</td></tr>
</table>

Remark: This subject fulfils CAR (STE) requirement.