COMPUTER SCIENCE
TEACHING AND LEARNING SUPPLEMENT

INDEPENDENT SCHOOLS TASMANIA
TASMANIAN CATHOLIC education office
Tasmanian Government

Teaching and Learning Supplement
COMPUTER SCIENCE 3 (ITC315118)

ADVICE FOR TEACHERS

This document helps to describe the nature and sequence of teaching and learning necessary for students to demonstrate achievement of course outcomes.

It suggests appropriate learning activities to enable students to develop the knowledge and skills identified in the course outcome statements.

Tasks should provide a variety and the mix of tasks should reflect the fact that different types of tasks suit different knowledge and skills, and different learning styles. Tasks do not have to be lengthy to make a decision about student demonstration of achievement of an outcome.

COURSE SPECIFIC ADVICE

This Teaching and Learning Supplement for *Computer Science 3* (ITC315118) must be read in conjunction with the *Computer Science 3* course document. It contains advice to assist teachers delivering the course and can be modified as required. This Teaching and Learning Supplement is designed to support teachers new to or returning to teaching this course.

SEQUENCE OF CONTENT

Computer Science 3 (ITC315118) is divided into four compulsory sections:

1: Problem Solving and Programming (70 hours)
2: Computer Fundamentals and Computer Limitations (40 hours)
3: Social / Ethical Issues and Professional Responsibility (10 Hours)
4: Computing Option (30 Hours)

These could be delivered through the sequence described in the following pages.

PROBLEM SOLVING AND PROGRAMMING (70 HOURS)

TEACHING AND LEARNING

Algorithm Design and Problem Solving	***Examples of learning activities***
	Learners:
	develop algorithms to describe rules within board games, TV gameshows, playground games or sports, and describe in pseudocode or flowchart form
	trace simple algorithms that are provided to them
	produce a wall chart of an algorithm to formally describe the processes that take place in a familiar situation e.g, operating a set of pedestrian lights, deciding which line to choose at a supermarket checkout, using a beam balance or kitchen scales to measure out a given mass, troubleshooting a failed light in a house, making a set total from a collection of different valued coins etc.
	develop algorithms for tasks commonly encountered in computing e.g., keeping track of the highest score and player in a game, calculating a running

total and average of a set of numbers being input, a simple search of a sorted collection of data, common sorting routines, validating a date of birth

identify and correct errors of logic in algorithms provided to them – for example, a flawed algorithm that describes presenting a bank card to an ATM and verifying the PIN, allowing for a fixed number of incorrect attempts

develop an algorithm for a situation that they will later code in Java – e.g., a simple game

produce and present a research report on commercially or socially important algorithms such as search engine algorithms, recommendation algorithms, trading algorithms, algorithmically-driven advertising, etc.

Produce an infographic that compares different sort algorithms in terms of speed/efficiency, resources needed and elegance

produce a simulation or test for comparing the performance of two algorithms for the same task

research and present findings on known limitations of some algorithms, for example cases where the "Greedy Algorithm" for making change with minimal number of coins does not produce the optimal solution.

Programming

Examples of learning activities

Learners:

use a provided piece of code to produce defined output – for example, using simple graphics operations to create a picture suitable for an animated book for 5 year olds

complete an applet development task in pairs using the Navigator/Driver approach (pair programming)

modify simple applets and code snippets to change their behaviour – for example, using simple graphics operations to modify an existing picture

extend simple applets and sections of code to perform additional tasks

create an applet based on screenshots or video of the required behaviour

review the code produced by their peers

take a piece of working code and introduce 3 syntax errors – the resulting flawed code is then swapped and learners find the errors in the code they receive

take a piece of working code and introduce 2 semantic or logic errors – the resulting flawed code is then swapped and learners find the errors in the code they receive

predict the output of pieces of code, without access to a computer to run it on

produce a trace table for a given piece of code under a set of conditions

produce a user interface based on sketches or photos provided to them

as a small team, develop an applet using versioning in a code repository

produce a screencast explaining an aspect of programming (for example the use of the IDE, the conventions of accepted program style such as indenting, naming, use of braces, brackets, parentheses, etc.).

SUPPORTING STUDENT RESPONSES AND ELABORATIONS

Providers and learners should make themselves familiar with the current Information Booklet that will be available to learners in the final examination. This booklet contains content that learners are expected to have studied but are NOT expected to have memorised. Use of the booklet during the course is encouraged in order to build fluency and skills while reducing the need for excessive memorisation.

Discussions amongst CSTA members has resulted in a set of pedagogical principles for teaching and learning programming. These may be of use.

They include

- encourage a Growth Mindset – abilities are not fixed, they can be developed,
- encourage learning from failure – failures are part of learning, and in programming the process of designing and testing is based on an expectation of a level of failure,
- Modify before Create – learners should be provided with starter code or code to be modified, before they are expected to create code from scratch,
- Concrete to Abstract – where at all possible, introduce concrete examples before moving to abstract concepts. For example, use fixed values in simple programs before developing the need to use variables,
- look for metaphor and simile – many concepts in programming have parallels in other situations. For example, learners might be asked to consider how they might complete "an array is just like a…". See https://www.sciencedirect.com/science/article/pii/S1570868308000463 ,
- Pair Programming – two people are more likely to see opportunities and errors. A common model is to assign roles to each pair or "navigator" and "pilot/driver". Only the pilot/driver touches the keyboard, but under the direction of the navigator, with roles swapping every 10 minutes or so.

WORK REQUIREMENTS

The equivalent of:

- Assessment tasks (10 programming tasks). Program development with accompanying explanation of specific aspects of the Java language.
- Assessment task – Algorithms (approximately 2 hours)
- Assessment task - SDLC (approximately 2 hours)

Fully documented programs (2 programs). These may form part of the Java language assessment tasks.

RESOURCES

All URLs (website addresses) cited were accessed and checked for accuracy and appropriateness of content on 20 Dec 2017. However, due to the transient nature of material placed on the web, their continuing accuracy cannot be guaranteed

Algorithm design and problem solving

CS Field Guide – Algorithms http://www.csfieldguide.org.nz/en/chapters/algorithms.html

Khan Academy – Algorithms https://www.khanacademy.org/computing/computer-science/algorithms

http://www.slate.com/articles/technology/future_tense/2016/02/how_to_teach_yourself_about_algorithms.html

Programming

General programming approaches

Debugging with a rubber duck https://rubberduckdebugging.com/

GitHub Student Pack https://education.github.com/pack and Classroom https://classroom.github.com/

The Computing Ed blog run by Mark Guzdial https://computinged.wordpress.com is a useful resource, though being US-based some of the resources are not as relevant to Australian learners. The academic papers by Guzdial and others are of high standard.

Java programming resources

Current and past Computer Science teachers have assembled a collection of resources which are available via a Dropbox link. New providers are encouraged to arrange to have access to this collection, and, along with other providers, to contribute sharable resources to this collection.

Java Cheat Sheet **http://introcs.cs.princeton.edu/java/11cheatsheet/**

(this is aimed at application development rather than applets)

Java Tutorials **http://www.tutorialspoint.com/java/index.htm**

Oracle Java Tutorials **https://docs.oracle.com/javase/tutorial/**

Java Programming video tutorials **https://www.youtube.com/watch?v=HI-zzrqQoSE&list=PLFE2CE09D83EE3E28**

Open Courseware courses on Java:

https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-092-introduction-to-programming-in-java-january-iap-2010/

Stanford CS101 videos **https://www.youtube.com/results?search_query=stanford+cs101**

Harvard CS50 videos **https://www.youtube.com/user/cs50tv**

Original Java code conventions **http://www.oracle.com/technetwork/java/codeconventions-150003.pdf**

Programming resources for this particular course include https://catscompsci.wordpress.com/java-practical-activities/ and https://www.youtube.com/user/CatComputerTeacher

COMPUTER FUNDAMENTALS AND COMPUTER LIMITATIONS (40 HOURS)

TEACHING AND LEARNING

Number representation	***Examples of learning activities*** Learners: compete in an online game where they have to represent decimal values in binary complete conversions between binary, decimal, hexadecimal for integers complete worksheets that include representation of positive and negative integers andfloating point numbers in binary construct working models of binary representation using LEDs or similar participate in physical tasks such as card-based binary counting, produce a CommonCraft-style video explaining two's complement representation and its purpose.
Data representation	***Examples of learning activities*** Learners: convert text messages into ASCII/Unicode and then binary manually, use online tools to convert between text formats and underlying binary,

develop an "encryption" method to allow text messages to be sent privately, based on simple reversible changes to the underlying binary representation,

use simple paper-based representations of images to explain pixel representation,

produce a short explainer using tools like Biteable to show how non-numeric data like colours, characters and instructions are represented.

Logic laws, truth tables and K maps

Examples of learning activities

Learners:

construct truth tables for logical expressions

use online tools to convert between logical expressions and truth tables

build simulations of logic gates using online tools.

simplify complex logical expressions

represent real-world logical conditions as logical expressions (for example, the legality of a 19 year old being able to legally consume alcohol depending on where within Australia, the US, Canada or Saudi Arabia they are)

construct and simplify Karnaugh maps by hand (on paper).

TOY

Examples of learning activities

Learners:

write simple TOY programs,

produce a wall chart that shows a TOY program and its corresponding representation in pseudocode and Java,

corrupt a working TOY program and the resulting programs are swapped within the class. Learners then attempt to debug the corrupted program to produce a working version.

SUPPORTING STUDENT RESPONSES AND ELABORATIONS

Providers and learners should make themselves familiar with the current Information Booklet that will be available to learners in the final examination. This booklet contains content that learners are expected to have studied but are NOT expected to have memorised. Use of the booklet during the course is encouraged in order to build fluency and skills while reducing the need for excessive memorisation.

Some students may find the mathematical nature of some of this content challenging. Provision of physical models and using some of the physical performance approaches from the CS Field Guide and CS Unplugged may prove useful in overcoming this challenge.

WORK REQUIREMENTS

The equivalent of:

- Assessment task - Logic laws, truth tables and Karnaugh maps (approximately 2 hours)
- Assessment task - Number representation(approximately 2 hours)
- Assessment task - Data representation (approximately 2 hours)
- Assessment task - TOY machine (approximately 2 hours)

These assessment tasks would typically include one or more problems to be solved, along with explanation of some aspect of the topic.

RESOURCES

Binary: integers, binary addition, signed integers, two's complement, binary addition. floating point representation

CS Field Guide http://www.csfieldguide.org.nz/en/chapters/data-representation.html

Tutorial videos https://www.youtube.com/playlist?list=PLADC99E719334 1529

Binary game https://studio.code.org/projects/applab/iukLbcDnzqgoxuu810unLw

Binary, Hex and Network Games https://samsclass.info/123/quiz/

Logic

Intro to Boolean logic http://www.i-programmer.info/babbages-bag/235-logic-logic-everything-is-logic.html

Wolfram Alpha logic widget – takes Boolean expression, provides truth table, logic circuit and Venn diagram http://www.wolframalpha.com/widget/widgetPopup.jsp?p=v&id=4c86f3bbcab249f879058d1825887571

Logic circuit construction simulator https://academo.org/demos/logic-gate-simulator/

Logic Circuit builder/simulator https://logic.ly/demo/ (requires Flash)

Fundamental logic gates – definition, corresponding Boolean expression, truth table, illustration, application https://www.wisc-online.com/learn/career-clusters/manufacturing/dig1302/basic-logic-gates

Interactive logic expression to truth table http://turner.faculty.swau.edu/mathematics/materialslibrary/truth/

Architecture

http://theteacher.info/index.php/f453-advanced-theory/3-3-3-architectures/notes/192-von-neumann-architecture-notes

TOY

https://introcs.cs.princeton.edu/java/62toy/

Sections of https://www.csie.ntu.edu.tw/~cyy/courses/assembly/07fall/lectures/handouts/lec06_toy_assembly_4up.pdf

General

Biteable https://biteable.com/

Commoncraft videos https://www.commoncraft.com/video-library

SOCIAL/ETHICAL ISSUES AND PROFESSIONAL RESPONSIBILITY (10 HOURS)

TEACHING AND LEARNING

Social issues ***Examples of learning activities***

Learners:

maintain a collection of media articles related to current social issues in computing,
engage in a class debate on a social issue,
undertake a role play based on a social issue.

Ethical issues ***Examples of learning activities***

Learners:

maintain a collection of media articles related to current ethical issues in computing,
use a simulation that represents an ethical issue (for example, Moral Machine),

engage in a class debate on an ethical issue.

Professional responsibility	***Examples of learning activities***

Learners:

use a professional code of conduct from another profession and identify parallels in the computing profession,
use an existing code of conduct from (for example) BCS or ACS to identify how to respond to ethical or social dilemmas.

SUPPORTING STUDENT RESPONSES AND ELABORATIONS

There are links between the Professional Issues and the Social and Ethical issues, and it is useful to make these links while at the same time identifying the nature of each type of issue.
Tasks based on current events, which require learners to identify the underlying issues and understand competing and conflicting viewpoints, can increase relevance. Ethical dilemmas can be examined via role play where students may be asked to take a view different to their own.

At the time of writing there were many commercial, educational and philanthropic organisations with strong interests in promoting careers in computing-related fields – these organisations typically release career guides, host career and industry sessions (face to face or online), provide video case studies etc.
Careers information is readily available from a range of sources, although it is worth noting that many areas of computing employment are in businesses that employ significantly fewer staff than the ABS definition of "small business" (<20 staff). The importance of self-employment, very small business and entrepreneurial/start-up culture need to be considered when looking at career possibilities in computing, especially in Tasmania.

WORK REQUIREMENT

The equivalent of:

Assessment task - Social issues, professionalism and ethics (1000 words)

RESOURCES

Ethical issues:

Australian Computer Society video playlists

Ethics: Untested System
https://www.youtube.com/watch?v=mugeCY3vbxo&list=PL8BUtM6njqLuhjFZGj1XLluQwNsIng6JW

Ethics: Early Launch

https://www.youtube.com/watch?v=v5M7ohdZ6qA&list=PL8BUtM6njqLsSqbpWzZ59bdy2IOmZXpQU

Ethics: Development Methodology

https://www.youtube.com/watch?v=0npm9cEJBWY&list=PL8BUtM6njqLvao3qQxNCT6nnQI0OVMDq2

Ethics: Overseas codeshop

https://www.youtube.com/watch?v=rBfJ07gfHyc&list=PL8BUtM6njqLv24V2Vm8xvNcIMUg-DkEqI

The EthicalCS organisation https://ethicalcs.org/ has classroom resources. The https://twitter.com/hashtag/ethicalCS?src=hash&lang=enTwitter community discuss ethical issues in Computer Science from time to time. A summary of some of the discussions is at https://medium.com/trytobegood/how-do-educators-integrate-ethics-and-social-justice-in-computer-science-curriculum-cf2c2f288e6b

MIT's Moral Machine, an attempt to gather public views on ethical issues http://moralmachine.mit.edu/

Social Issues:

It is worth monitoring news media and social media for current social issues that are relevant to the learners. There are some historic collections of social issues such as http://socialissues.cs.toronto.edu/ (a dated blog which attempts to update a book on this topic from 1973) which serve to indicate that some topics like computerised voting, the participation rates of women in computing, privacy and security, AI, open vs closed systems and data, etc., are issues that remain with us long-term.

Careers and career pathways

ACS Women in ICT Careers videos:

https://www.youtube.com/watch?v=cZ69qAsZxgw&list=PL8BUtM6njqLuWFp7jdTgpJMZadK3PP0sf

Industry reports on the ICT industry in Tasmania and Australia are useful in conveying the employment (as distinct from career) situation. Sources include ACS, TASICT, State and Federal Government reports.

COMPUTING OPTION (30 HOURS)

TEACHING AND LEARNING

Computing Option	***Examples of learning activities***
	Learners explore a computing topic in depth. The topic could include, for example: production of a Java applet or application for a client following the software development lifecycle Object-oriented programming in other languages game development in a suitable environment exploration of network programming programming for mobile devices Media Computation cryptography, compression and security artificial intelligence and machine learning human-computer interaction computer forensics ethical and legal aspects of computer science application of computer science principles to another field (e.g. life sciences, psychology, law) big data and data science exploration of Java libraries exploration of alternative Java programming environments (e.g. Greenfoot, Robocode) programming LEGO® robots using LeJOS programming of embedded systems and microcontrollers digital electronics.

SUPPORTING STUDENT RESPONSES AND ELABORATIONS

Providers and learners should make themselves familiar with the current Information Booklet that will be available to learners in the final examination. This booklet contains content that learners are expected to have studied but are NOT expected to have memorised. Use of the booklet during the course is encouraged in order to build fluency and skills while reducing the need for excessive memorisation.

The Computing Option can provide an opportunity for learners to pursue a specific interest or skill. It can also be a way for learners to build and refine their understanding of the Java programming language, which may be a preferred approach for some providers. In some cases, a provider may suggest that many or all learners undertake the same option.

Computing Option topics could include new and emerging areas of computing, allowing learners to undertake significant study in contemporary areas.

Before approving a learner to embark on a specific topic for the Computing Option, providers must ensure that the topic will allow evidence to be collected for:

> both Criterion 8 (apply personal skills to plan, organise and complete activities) and Criterion 9 (communicate technological information)
>
> and
>
> either Criterion 1 (design, extend and improve algorithmic solutions to a range of problems) or Criterion 6 (apply the software development life cycle to a variety of problems),
>
> along with at least one of the remaining criteria.

It is suggested that Computing Option be introduced to learners earlier in the course rather than later, as learners may well wish to commit more time to it than anticipated. However learners need to have studied the relevant content prior to commencing the Computing Option – for example if learners are undertaking a major Java project they need to have completed the Programming section beforehand.

Professional judgement is needed to ensure that learners undertake an option that is achievable, has suitable sources of support, and is not reliant on factors outside the learner's control. Learners should be regularly reminded of the self-management expected, and supported in developing the skills required for the Criterion 8 component of the Computing Option.

WORK REQUIREMENTS

Option Product: documented program, or report on investigation (4000 words or equivalent)

RESOURCES

A simple project management tool may prove useful to assist students to meet deadlines and manage their resources. Many online PM tools have a free version for education or for small projects. There are new products emerging constantly, but providers could consider possibilities ranging from simple document management tools such as Office365/OneDrive or Google Drive through to dedicated PM tools.

The best choice depends largely on the needs of the particular situation and providers should scan for appropriate tools.

Zoho Projects https://www.zoho.com/projects/ free version may meet most needs
GitHub Education https://education.github.com/ offers a free code management repository for students undertaking a programming project.

It may be worthwhile introducing students to Design Thinking if they are not familiar with it.
https://www.interaction-design.org/literature/topics/design-thinking and https://www.ideou.com/pages/design-thinking can assist this process.