INFORMATION TECHNOLOGY (Updated Sep 2014)

CONTENTS:

OVERVIEW OF INFORMATION TECHNOLOGY

Refer to page 7 of the CAPS document for a graphical representation of the six main topics of Information Technology namely:

1. Systems Technologies (page 14)
2. Internet & Communication Technologies (page 13, 15)
3. Social Implications (page 17)
4. Data and Information Management, solution development (page 11, 16)

A brief summary of each topic is given on page 8 of the CAPS with detailed descriptions for each topic listed next to each bulleted point above.

A. MEANS OF ASSESSMENT

Paper 1 (Theory)		3 hours	180 marks reduced to 100	
Paper 2 (Practical)		3 hours	120 marks reduced to 100	(200)
Practical Assessment Task	PAT			(100)
School Based Assessment	SBA			(100)
TOTAL				(400)

B. **REQUIREMENTS**

INFORMATION TECHNOLOGY PAPER I (THEORY)

The theory examination will cover questions on the following sections:

1. Systems Technologies
2. Internet & Communication Technologies
3. Social Implications
4. Data and Information Management & Solution Development

Weighting of Sections	
1. Systems Technologies and Terminology	45 marks
2. Internet & Communication Technologies	55 marks
3. Social Implications	20 marks
4. Data and Information Management & Solution Development	60 marks
TOTAL	180 marks reduced to 100 marks

INFORMATION TECHNOLOGY PAPER II (PRACTICAL)

Solution Development, Data and Information Management

The Practical Examination will cover questions on databases (SQL), algorithms and object oriented programming (OOP). The candidate may use a text based interface or a Graphical User Interface (GUI) in the practical exam. Databases will either be assessed using an application program OR via SQL statements using a programming language.

Weighting of Sections	
4. Data and Information Management and Solution Development	
TOTAL	120 marks reduced to 100 marks

PRACTICAL ASSESSMENT TASK (PAT)

The PAT is to be completed during the course of most of the Grade 12 year and should be formatively assessed so that the candidates have the opportunity of submitting their best work.

Content to be covered: Programming Project 100 marks

Weighting of Sections	
4 Data and Information Management and Solution Development	100 marks
TOTAL	100 marks

SCHOOL BASED ASSESSMENT (SBA)

These Subject Assessment Guidelines must be read in conjunction with the IEB Manual For The Moderation of School Based Assessment (SBA) 2012.

All schools must make available the SBA evidence of all learners should it be required by the IEB or Umalusi.

The SBA assessment comprises 25% of the total assessment for the National Senior Certificate. The requirements for the school based assessment component are outlined in the following table:

SBA COMPONENTS	
SQL Test (4)	10%
Normalisation Test (4)	10%
OOP Test (4)	10%
Theory Test (1, 2 and 3)	15%
Data Aware Components (4)	15%
Preliminary Examination (1, 2, 3 and 4)	40%
TOTAL	100%

SUMMARY OF MARK DISTRIBUTIONS
(These marks are approximate and act as a guideline)

	1	2	3	4	TOTAL
Paper II				*120 marks reduced* to 100 marks	100
Paper I	*45 marks adjusted to* 25 marks	*55 marks adjusted to* 40 marks	*20 marks adjusted to* 15 marks	*60 marks adjusted to* 20 marks	100
PAT				100 marks	100
SBA	15 marks	20 marks	5 marks	60 marks	100
Total	**40 marks**	**60 marks**	**20 marks**	**280 marks**	**400**
Weighting	10%	15%	5%	70%	100%

Taxonomy for Information Technology Practical Assessment

Level	**Weight**	**Description**	**Examples**
1	30%	Routine procedures, Syntax	• Variable declarations • Variable assignment • Use of constants • Operators within calculations • Importing additional functionality • Declaration of classes as prompted by a question • Declaration of typed and non-typed methods (functions and procedures) under direction • Simple I/O • Simple Selection and Iteration (non-nested) • Array declaration and usage (1-dim) • Maths functions (round, sqrt, etc.) • Type casting and conversions • Boolean operations (AND, OR, NOT) • Simple non-nested control structures (if, switch, for, do ... while, while) • SQL: single table, query with specified fields and either a single restriction (no Boolean operators) or sorting

Level	Weight	Description	Examples
2	40%	Multi-step procedures	• Application of nested loops and if statements in taught algorithms • String functions (length, extraction, searching within) • Declaration and instantiation of objects • OOP modelling under direction: method declarations (simple accessor, mutator methods), few parameters and returns • Learned (generic) algorithms applied to simple situations • Complex tasks using multiple 'simple application'-type skills (e.g. string manipulation and sort, or multiple elements of string manipulation) Adapting learned algorithms to new scenarios (specific algorithms) • SQL: Simple queries on joined tables (using WHERE) • SQL: Use of aggregate functions • SQL: Use of functions in SQL • SQL: statements using two of joined tables, SQL functions, more than two logical operators
3	30%	Problem Solving	• Unseen/unprepared problems requiring students to adapt learned algorithms significantly, or to create a unique solution to a problem. • OOP modelling from problem definition, unguided • Problems focusing on efficiency and code elegance (e.g. 'marks will be awarded for efficient solutions'). • SQL: statements involving more than two of joined tables, SQL functions, multiple logical operators, grouping (with or without GROUPs)

Taxonomy for Information Technology Theory Assessment

Level	Weight	Description	Examples
1	30%	Knowledge, Comprehension,\	• Questions where the learner is required to recall knowledge from the Grade 10, 11 and 12 IT syllabus: Recall, define, explain, identify, describe, match columns, label.
2	40%	Application and Analysis	• Compare two solutions and give a recommendation (simple scenario). • Questions where the learner analyses a scenario and makes justified suggestions: • Suggest a network architecture given the requirements of users. • Questions where the learner is required to apply existing knowledge to a given scenario • List hardware and software specifications given a user's requirements.
3	30%	Synthesis Evaluation, Creativity, Problem solving	• Analyse an algorithm and suggest improvements. • Analyse or compare data structures. • Compare two solutions and give a recommendation (scenario which requires considerable in-depth thinking). • Questions where the learner must evaluate or make a judgement based on given criteria or content: • Analyse the given information and provide an opinion that is substantiated with the correct facts. • Create UML models / classes from a given scenario

C. INTERPRETATION OF REQUIREMENTS

DETAILS OF PAPER I – THEORY EXAMINATION

The details for the content to be covered can be found in Appendix G of this document.

DETAILS OF PAPER II – PRACTICAL EXAMINATION

In the three hours of the examination, candidates will be required to develop solutions for one or more unrelated problems, each of which may have more than one section. The practical exam will only examine databases (SQL), algorithms data structures and Object Oriented Programming concepts.

The candidates will be examined on the skills and content described in the CAPs document.

Examination Time

Additional time for printing is allowed over and above the three-hour solution-development time. Conditions permitting, the candidates should print their own work on completion of the three hour examination, otherwise it is the teacher's responsibility to ensure that all required work is printed out, without any alterations taking place.

Data CDs supplied by the IEB

The Information Technology teacher is required to check the data on the CD provided by the IEB to schools the day BEFORE the examination is to take place. The Information Technology teacher should ensure that the CD is virus-free and can be accessed and read by the appropriate software. The Information Technology teacher is also required to make arrangements to make the data available to the candidates writing the examination as outlined in the examination instructions that accompany the question papers. These instructions are delivered in a separate envelope and are to be opened 2 days before the examination is written.

Procedures for the marking of the Grade 12 Information Technology practical examination

- The IEB will send Information Technology teachers an interim marking guideline and an interim marking scheme together with the practical examination question papers.
- Information Technology teachers are to use the marking guidelines and the marking scheme to mark a selection of their learner's scripts (no more than 5) within two days of the examination.
- During a period of two to five days after the examination, all the cluster groups should meet in their region under the chairmanship of their cluster leader. The cluster groups that are in the same region as the examiner do not have to meet at this stage. The purpose of these meetings is to discuss potential problems and clarify issues with regard to the marking of the practical examinations. The consensus recommendations for marking the practical examinations, agreed at these meetings, together with the cluster group leader's name and cluster should be submitted to <iebcompza@yahoogroups.com> and the Assessment Specialist at the IEB.

- Seven days after the examination, the practical examiner will hold a meeting with teachers from the cluster groups in the examiner's region. The marking of the practical paper will be standardised at the meeting with the examiner using the input from the cluster groups and the input from the teachers present at the meeting.
- The marking guidelines and the marking scheme will be adjusted by the examiner after the meeting and sent to <iebcompza@yahoogroups.com> and published on the IEB website.
- Teachers are to mark their candidates' practical examination scripts using the revised marking guidelines and revised marking schemes.
- One of the tasks of the moderating committee will be to monitor the marking of the practical examinations against the revised marking guidelines.

Submission of the practical examination for moderation

The marked practical examination scripts for all candidates must be submitted to the IEB by 15 November. Teachers must submit all scripts in **one envelope** labelled with the following information:

- The clearly written heading 'Practical Examination Scripts: Information Technology';
- The centre number of the school.

The envelope must contain:

1. ALL marked practical examination scripts of all candidates registered for IT. The teacher should also make clear their reasons for deducting marks on the mark sheets to facilitate moderation.
2. a single CD (labelled 'Examinations' with the centre number) containing each candidate's work in a separate folder with their examination number as the name of the folder
3. a copy of the revised marking guidelines and marking scheme the teacher used (with any justified alterations or annotations)

DETAILED REQUIREMENTS FOR SCHOOL BASED ASSESSMENT (SBA) AND THE PERFORMANCE ASSESSMENT TASK (PAT)

The SBA and the PAT will be externally moderated and each candidate is required to submit all the assessments in October of their Grade 12 year. They should be informed of these requirements towards the end of their Grade 11 year and supplied with detailed task descriptions and the criteria against which they will be assessed.

Additional information

Teachers are also strongly advised to join iebcompza@yahoogroups.com and email their questions to this address. The procedure for becoming a member of this group is available from the IEB Assessment Specialist responsible for Information Technology.

Mark Sheets

At the end of this document there are example mark sheets for the Data Aware Components Task and the Performance Assessment Task (PAT). Teachers may adjust these mark sheets, but the mark allocations must remain similar.

Formative Assessment

All informal tasks in the SBA and the PAT must be formatively assessed. When teachers review a task, they should listen to the candidate and give advice. They should be careful not to give the candidate the solution. They should suggest alternate resources and query explanations. This formative assessment is designed to help the candidate learn. It also creates the opportunity for the teacher to monitor progress, give additional input and helps guard against plagiarism. A task that has been poorly managed by the teacher can lead to substandard work by the candidate.

The programming project must follow the System Development Life Cycle (SDLC) it gives the candidate a structured manner to approach large projects and the documentation provides evidence of the candidates process in developing the product. Ensure that all planning, designs and algorithms are done prior to actual coding and implementation. (<http://courses.cs.vt.edu/csonline/SE/Lessons/Waterfall/index.html>)

The Programming Project should be completed over a number of months and must be closely monitored to avoid plagiarism.

Task Descriptions

The tasks should be detailed and follow the principles of assessment. The tasks should be descriptive, allow for formative assessment and give details of deadlines and how the task is to be structured. The task must give the candidate all the information required to help them produce their task. The task and the rubric must be moderated at cluster level using the principles of assessment.

Marking and Moderation

The contents of the SBA and PAT will be marked internally and moderated according to the IEB's moderation process. Refer to section D of this document for further details in this regard.

SCHOOL BASED ASSESSMENT (SBA)

The requirements for SBA are as follows:

- SQL Test (4)
- Normalisation Test (4)
- Data Aware Components Task (4)
- OOP Test (4)
- Theory Test (1, 2 OR 3)
- Preliminary Examination (1, 2, 3 and 4)

The Normalisation Test, Theory Test, SQL Test and OOP Test must be set to cover the Assessment Standards described in the SAGs for Grade 12. The preliminary examination must be set with the same weightings as those described for the November practical and theory examinations. These examinations must be set to different cognitive levels.

SQL TEST

This test should take approximately 45 minutes and must include some advanced SQL statements using more than one table, JOINs and aggregate functions. 60% of the paper should include simple SQL statements based on one table.

NORMALISATION TEST

This paper can be written as a theory or practical paper, but no programming is allowed. This test must involve normalising a single large table into more than one table. Do not go beyond 3NF (Third Normal Form). Include definitions of the types of normal forms, derived data, keys, anomalies, repeating groups and other normalisation terms. Candidates should be able to apply these definitions to the given table(s).

OOP TEST

This test is to be written as a theory test in class, and should be about 40 – 60 minutes in length. The content may vary, but it is recommended that the following are tested:

- Fields and methods of an object,
- Constructor, accessor and mutator methods,
- toString method
- OOP terminology such as information hiding, inheritance (not for 2014), polymorphism (not for 2014), overriding (not for 2014), overloading,
- An application class to use the object definition class
- Methods or algorithms within the application or object class
- Typed and void methods
- Types of secondary storage for data such as a database or text file and related code/algorithms to store and retrieve data
- Compare/analyse/application of different types of data storage
- Data structures such as arrays, parallel arrays and arrays of an object
- Compare/analyse/application of different types of data structures

THEORY TEST

This test should be about 40 – 60 minutes in length and should contain a range of questions covering content from sections 1, 2 OR 3. A range of cognitive skills must be assessed as outlined in the taxonomy for Theory Assessment. It is not advisable to use a full scale exam for this test as it penalises the candidates.

DATA AWARE COMPONENTS TASK

The purpose of this task is to familiarize students with the concept of data aware components relating to the fields and tables of a normalised database. Since Data Aware Components will not be examined in the Practical examination, this task will serve to equip learners with some experience in the developing of data aware components in their user interfaces.

Each learner must provide their own topic for their task which must differ for each learner and for each year.

The learner must define a database with a **minimum** of two tables that has a one to many relationship between the tables. The database must be normalised to include primary and foreign keys. The learner must create a programming project with at least 4 Graphical User Interfaces screens. One screen must display the entire table in the database. The second must display a single record of a table in separate fields. These fields must be Data Aware with components that will actively manipulate the update the fields using the data stored in the fields of the database. This screen must have components to:

- Go to the first record
- Go to the next record
- Go to the previous record
- Go to the last record
- Insert a record
- Delete a record
- Search for a record and update a record
- Edit a record

When the user clicks any one of these components, the fields must respond accordingly.

For each table in the data base, there must be the above two screens (one to show the entire table and one to show record by record for the same table). If the learner has two tables in the database, then there must be four screens.

PERFORMANCE ASSESSMENT TASK

Programming Project

The programming project represents the development cycle of a product. The purpose is to give the learner a meaningful experience of a larger project and development cycle than is usually possible in a classroom situation.

Scope

This project should be more complex and include more features that those of a practical examination paper. It should not be something that the learners can produce in as little as a week. Inappropriate examples might include a calculator or a currency converter.

Topic and Content

The program must be written in one of the programming languages approved by the IEB. (See the relevant IEB circulars with regard to the languages currently approved by the IEB for use with practical work and this programming project.)

The program needs to meet the following criteria:

- **User friendly** –
  - **Interface** – the interface must be user friendly (preferably a GUI), easy to use and task appropriate. If the program does not use a GUI there has to be a very good, task appropriate reason for this (which is adequately explained in the design document). If a command line utility is envisaged then the 'engine' must be accessible either through a command line or a GUI interface. In short, there is virtually NO application design that should not include a GUI element.
  - **Data Flow/program operation** – the learner must ensure that the sequence of steps required to use the program and complete a task are clear, easy to follow and logical.
- **Storage/Data persistence** – data must be stored and retrieved from session to session (this can be in the form of conventional files OR a database OR both). Database work is to be encouraged, as is the use of SQL. The storage is appropriate to the program (a game should have the ability to save and load a game and have high scores, etc.)
- **Separation of interface and engine –** the 'working code' **must not be embedded in the interface** (i.e. it must be in separate classes/units). Communication between the interface and the working code is in the form of parameters and typed methods (functions). A limited amount of code in the interface is acceptable only if suitably justified in the planning. This is essentially a design pattern called *Model View Controller*.
- **Good data internal structures** – There has to be some form of internal representation of data (i.e. classes/records/arrays – or any combination of these). Data structures must be logical and task appropriate. Classes are to be encouraged.

Refer to the CAPS document for a description of the programming skills required for the practical examination. These serve as a guide as to the **minimum** level of programming skill that should be demonstrated in the programming project.

To be fair to the learner and to ensure a good product at the conclusion of the project it should be completed (and assessed) in 5 phases.

Phase		Description	Marks
What must be submitted by the candidate:	Project Specifications	List (and describe) the functions that your program needs to achieve in order to be a 'success'.	14
	Design Document	Design the user interface, sequencing (data flow), class and persistent storage of the program in detail.	31
	Coding and Technical Document	Write the program following good programming techniques and document it by printing the code and explaining critical algorithms.	50
	Testing Document	Document what is to be tested, the test data used and the results of the testing.	5

In order to give the candidates a starting point, **teachers may provide templates (provided at cluster/regional/national level)** that show the type of content required in the ***Project Specifications***, ***Design***, ***Technical*** and ***Testing*** documents. **Strongly** encourage them to use these templates as it will drastically reduce their workloads, standardise the PAT and greatly simplify your marking/moderation.

Mark Sheet

See Appendix B on page 17 for an example.

D. THE MODERATION OF THE SBA AND PAT

The IEB Manual for the Moderation of School Based Assessment for the Senior Certificate Examination describes generic procedures. The subject specific requirements and procedures described in this manual supply the additional subject specific detail beyond that in the generic manual.

SUBMISSION OF SBA AND PATS

Regional Moderation

Schools will be assigned a regional moderator. Regional moderation of portfolios will take place between 15 September and 15 October. Regional moderators will liaise with their schools regarding how and when the moderation will take place and will choose a random selection of portfolios to moderate.

Where portfolios meet the required standards, it may not be necessary to send them for National moderation. However, both non-compliant and exemplary portfolios may be called for further moderation by the National panel.

For clusters where no regional moderator has been assigned, portfolios will automatically be moderated at National level.

National Moderation
Marked SBA and PATs must be submitted to the IEB by 31 October. Teachers must include

- The relevant documentation (see the IEB Manual for the Moderation of School Based Assessment for the Senior Certificate Examination distributed by Circular.
- the list, provided by the IEB, of candidates identified for sample moderation
- Their reasons for deducting marks to facilitate moderation. They should do this on the mark sheets.
- A teacher portfolio together with evidence of moderation of tasks and rubrics
- SBA of the candidates identified for sample moderation and any additional SBA that you wish to send should the sample not give a good representation of the group
- The PATs of all the learners
- A CD* (labelled 'SBA and PAT' with the centre number) for all candidates identified for sample moderation. Each candidate's work should be in a separate folder. The candidate's examination number should be used as the name of the folder. The moderation committee moderates a smaller sample of the work produced by candidates against specific criteria and is required to submit a report that is returned to schools.

The moderation committee uses the completed forms submitted after cluster moderation to guide their work and this committee has the authority to make motivated recommendations for the changing of marks should they deem this to be necessary.

1. APPENDIX A

NATIONAL SENIOR CERTIFICATE EXAMINATION

INFORMATION TECHNOLOGY

DATA AWARE COMPONENTS MARK SHEET

The purpose of this task is for candidates to demonstrate their ability to create a GUI with components that are bound to the fields in a database.

Criteria	Description	Possible Mark	Actual Mark
Normalisation The candidate has successfully designed a database with two tables linked by a one to many relationship	0 – Design is not normalised 1 - Design has two tables with no relationship or primary keys 2- Design has two tables with primary keys but not linked 3 - Design has two tables linked with an inappropriate primary AND foreign key 4 - Design has two tables linked with an inappropriate primary OR foreign key 5 – Design has two tables linked with an appropriate primary AND foreign key	**5**	
Create Statement for Table 1 List the SQL statement for the first table including the primary key.	0 – Incorrect or no SQL statement 4 – SQL statement is correct with primary key defined Deductions to a maximum of 4 (-1 for each error in declaring table and fields) (-2 if no primary key)	**4**	
Create Statement for Table 2 List the SQL statement for the first table including the primary key and foreign key	0 – Incorrect or no SQL statement 4 – SQL statement is correct with primary key defined Deductions to a maximum of 6 (-1 for each error in declaring table and fields) (-2 if no primary key) (-2 if no foreign key)	**6**	
Variety of Types Used In the tables there are at least one string(text), integer, real, boolean and date/time field.	Deductions to a maximum of 5 (-1 for each type of missing field)	**5**	
Programming Project GUI Each table is shown as two GUIs – one GUI to display the entire table and one GUI to display the table record by record giving a minimum of 4 screens.	4 – one mark for each GUI (-1 for each missing table)	**4**	
Naming Conventions Labels, buttons, tables, fields all correctly named according to conventions.	Deductions to a maximum of 2 (-1 for each incorrectly named field)	**2**	
Screen Design of the GUI Each screen is clearly laid out, with consistent fonts, colours, size of fonts, and buttons. Fields aligned and clearly labelled.	Deductions to a maximum of 8 (-1 mark for each layout error) Only penalise once for each type of error.	**8**	
Buttons Linked to Components The following buttons update the records in both of the record by record screens of each table: First; Previous; Next; Last; Insert; Edit; Delete and Search	For each button: 2 marks if the button updates successfully. 0 if it does not.	**16**	
TOTAL		**50**	

2. APPENDIX B

NATIONAL SENIOR CERTIFICATE EXAMINATION

INFORMATION TECHNOLOGY

PERFORMANCE ASSESSMENT TASK – EXAMPLE MARK SHEET

Project Specifications				
CONTENT				
Summary			**2 MARKS**	
0 – No summary or completely inadequate	1 – Summary partially done	2 – Summary encompasses all aspects of the problem		
Specifications of Program Function			**3 MARKS**	
0 – No Functions listed	1 – Function list is a single line statement/a list of 4 or less points	2 – Function list is a substantial list of appropriate outcomes	3 – Function List is complete and detailed	
Specifications of User Interface			**2 MARKS**	
0 – User Interface not specified or incorrectly specified	1 – One or two items are inadequately specified	2 – User Interface completely specified		
Specification of Help			**2 MARKS**	
0 – Help not discussed	1 – Help partially discussed with omissions and/or errors	2 – Help completely detailed including menu options and types of help available. Context provided for each help screen as well as storage of help related data has been specified		
Specifications of Data Storage			**3 MARKS**	
0 – No information given on the data to be stored	1 – Only a few items are incorrectly described	2 – Many items are described with a few errors	3 – All data to be stored has been correctly described	
Hardware and Software requirements			**2 MARKS**	
0 – Hardware and Software not discussed	1 – Hardware and software is partially discussed for development, includes detail for what software is needed for what task	2 – Hardware and software is completely discussed for development, software list includes versions		

System Design Document

Taking the template as an example, this document should be around 14 – 20 pages (including title page and table of contents). Its main task is to detail the actual design elements of the program, namely:

- User interface design (what the screens look like and what happens on them)
- Program flow (how the program works – linked to the interface)
- Class design (what the classes are, their fields and methods)
- Database/Storage design (what the persistent storage structure is)

System Design Document			
User Interface Design NB: The GUI screen can be designed in a RAD environment (e.g. NetBeans/Eclipse/Delphi), on paper, or in a graphics program like Paint. Screen mock-ups are possible without writing code, therefore screenshots are acceptable as evidence of design, All data to be displayed on the screen must be listed. The action elements on the screen must be listed and clearly described.		**8 MARKS**	
0 – 2 – No screen design evident/Screen design is cursory	3 – 5 – Screen design is evident but no consideration has been given to good design principles for an effective GUI or inadequate description of on-screen action elements or no indication of progression between screens	6 – 8 – Screen design present, layout good, all on-screen action elements tabulated and described in detail. Data access and error-checking are indicated for all action elements.	
Sequencing – (also known as What Happens When) **In this section you describe the flow of events in the program – planning this can make your program easier/more logical to use (can help you decide on interface elements such as wizards, etc.)** **NB:** The template contains flowcharts. The candidate may use any algorithmic representation of sequencing of events in the flow of the program such as pseudocode.		**5 MARKS**	
0/1 – No sequencing evident/Sequencing is rudimentary or cursory with little detail and large leaps of logic evident	2/3 – Sequencing is substantial but still shows leaps of logic/areas that have not been covered in appropriate detail	4-5 Sequencing is broken into sections to cover all aspects of the functions and features in the Specification document. Flow is clear, well represented and easy to understand. No logic gaps are evident.	
Class Design **The candidates must provide their class design and explain their choice of classes, fields and methods.** **NB:** The template contains tables and this structure must be used for the class design where each field and method is explained.		**8 MARKS**	
0 – 2 – No class design evident/class design is rudimentary or cursory with little detail. Fields are incomplete, methods are minimal/not well thought out/not well described.	3 – 5 – Class design is substantial but still shows obvious gaps in missing fields/methods. Method descriptions more thorough but elements still missing.	6 – 8 – Class design is thorough – all fields and methods are present and well described. Sub-methods are present. Methods and fields clearly relate back to the Specification document.	

Persistent Storage Design **The candidate must provide their storage design** **NB:** Storage design should be done in tables. Screenshots of tables with record structure and field types from database software can be acceptable.		**6 MARKS**	
0/1 – No storage design evident/storage design is rudimentary or cursory (e.g. 'uses a database').	2 – 4 – Storage design is substantial. Some record design and description of fields are evident. Descriptions are, however, cursory and show evidence that they have not been completely thought through. Not all files/tables/relationships covered.	5/6 – Record structure is described – fields are listed, typed and described. Data structure for text/typed files is described. Storage design is appropriate to purpose and matches the Specification document requirements.	
Explanation of Storage Design **The candidate must provide an explanation of their storage design** For example a text file may have been a better solution that a database as the data to be stored is small in value and simple. In this section you describe the way that data is stored so it can be re-accessed when the program is used again. Appropriate storage is what is required. DO NOT mark for quantity. What we need to see is that if, for example, a game is coded then high scores and save games are needed – and maybe other file handling to load appropriate data. DOES NOT have to be database.		**4 MARKS**	
0/1 – No explanation is given about storage or no understanding of the storage design the storage is shown.	2/3 – Explanation is substantial but it is not completely justified and there are some areas of confusion or lack of understanding of the implication of the storage design.	4 – Explanation shows in-depth understanding of the implications of the storage design and is completely justified.	

CODING and Technical Documentation

Taking the template as an example, this document can be anything from 10 – 100+ pages depending on the complexity and extent of the code that the candidate has written. Emphasis must be placed upon:

Comments for all the methods (these can be copied and pasted from the Design Document)	Good use of persistent storage
Separation of UI from working code	Defensive programming
Communication using typed methods (functions) and parameters	Fulfilment of Design Specifications
Good general programming techniques (naming, indentation, appropriate data structures, etc.)	User Experience

CODING NB: This is assessed by examining the actual code – no attention need be paid to documentation/layout/etc.			
COMMENTS		**6 MARKS**	
0 – code has not been commented	1 – 3 – Code contains some comments. Not all methods are commented. Comments are brief and contain little relevant detail. Scale the mark on the detail and quantity of appropriate comments.	4 – 6 – Code has been commented - 4. All methods have comments describing what they do, 5. Comments include the data they return (for typed methods) and the data they receive (parameters) 6. Steps in algorithms and solutions are commented as well. Variations of this will produce a different grade.	
Separation of UI from working code		**5 MARKS**	
0 – No separation – all code in the interface class/unit.	1 – 3 – Some separation. There are separate classes/units but work is still done in the UI. Insufficient further breakdown and separation of different aspects of the engine. This includes SQL statements for database centric programs (SQL is separation – you are passing off complex data handling to the database engine).	4/5 – Complete separation. Different classes are separated as well as the engine from the UI. The engine can be 'plugged into' a different UI that uses all the methods appropriately and will work without any issues.	
Inter-Code communication (Typed Methods and Parameters)		**5 MARKS**	
0 – No inter code communication (no typed methods (functions) or parameters.	1 – 3 – Some use of parameters/functions. Marks can be deducted (–1 per error type – multiple instances of the same error do not accumulate deductions) Errors include: Shows errors in comprehension of the concepts– unnecessary use of parameters, incorrect parameters types, parameters specified but not used, incorrect functions types, failing to return values in functions, failing to use the results returned by functions, using variables/fields where the value is best returned by a function.	4-5 Effective and conceptually correct use of parameters and typed methods (functions).	
Good General Techniques		**5 MARKS**	
0 – No techniques.	1 – 4 – Errors in techniques (–1 per error type – multiple instances of the same error do not accumulate deductions). Errors include: No indentation, single level indentation, inconsistent or inaccurate indentation, variable names do not clearly indicate what the variable is used for, multiple variables used instead of arrays, multiple if statements instead of switches, repetition of code (instead of using a procedure/function), code extending beyond the edge of the readable printed page.	5 – Technically perfect. Indentation immaculate. Variable names, data structures, etc. all correct.	

Persistent storage/Querying			**6 MARKS**	
0 – No persistence. Data is not saved or retrieved.	1 – 2 – Storage is utilised but is either minimal or badly implemented. Inappropriate storage structure selected, poor implementation, storage intrudes on workflow of user, storage implementation is trivial.	3 – 4 – Storage effective and appropriate to the nature of the task. Storage management is not intrusive and does not disrupt the workflow of the user.	5 – 6 – Storage is non-trivial and is implemented in a meaningful way. Includes i/o exception handling and meaningful error messages.	
Defensive Programming (data validation, exception handling, error messages)			**4 MARKS**	
0 – No data validation	1/2 – Cursory data validation/error trapping. Only focussed on limited areas of code (file handling). UI elements are poorly selected (i.e. do not include UI that prevents incorrect data entry). Important data type checking not implemented.	3 – Aspects are complete – potential major IO errors protected with exception handling, GUI elements used, potential maths errors trapped. There are, however, a few areas where the candidate has not implemented defensive programming. Error messages not as clear as they could be.	4 – All appropriate data is controlled and validated using code and appropriate GUI elements and exception handling. All error messages are descriptive and easy to understand.	
Fulfilment of Specifications **NB:** this can only be assessed by running the compiled program.			**6 MARKS**	
0 – Not achieved.	1 – 3 – Basic implementation of specifications. Obvious omissions in missing functions/significant amount of functions do not work as specified.	4 – 90% of specification achieved. Perhaps all functions are there but do not all work correctly OR almost all functions are there but those that are there work 100%.	5/6 – All specifications complete and working 100%	
User Experience **NB:** this can only be assessed by running the compiled program.			**4 MARKS**	
0 – Program does not execute.	1 – The user is lost – does not know where to start or how to achieve anything when using the program.	2/3 – Most of the program has a good user experience but navigating to some screens/functions is unnecessarily complex/impossible. Any aspect of the design/interaction is confusing or unsatisfying.	4 – An easy to use, completely easy to understand and to navigate program: a wonderful user experience.	

Technical Documentation				
CONTENT				
Externally Sourced Code **NB:** This must be present even if the candidate only declares that no external code has been used.			**1 MARK**	
0 – Not Present	1 – Candidate has declared used code. Confirm this with interview incorporating oral review of code and techniques.			
Explanation of Critical Algorithms **NB:** The core algorithms that are critical to the correct functioning of the program. There may be only a few (or even only 1) of these.			**3 MARKS**	
0 – Not Present	1 – 2 – Algorithm present with no description of significance/poor flowchart/pseudocode.	3 – A good, clear description of why these algorithms are critical. Good flowchart/pseudocode.		
Advanced Techniques **NB:** This must be present even if the candidate only declares that no advanced techniques have been used.			**5 MARKS**	
0 – 1 Not Present	2 – 3 – Spurious – candidate is manufacturing advanced concepts/not clearly explained	4 – 5 A good explanation of what the candidate regards as the 'extra mile' that they included in the project.		

TESTING Document			
TEST PLAN AND RESULTS		**5 MARKS**	
Testing has been planned using well selected or generated data. Data generated/selected for common, extreme and erroneous cases inputs Table/screenshots of tested data showing results of testing with common, extreme and erroneous cases	2 3		
		100 MARKS	

3. APPENDIX C

NATIONAL SENIOR CERTIFICATE EXAMINATION

INFORMATION TECHNOLOGY

NATIONAL/REGIONAL/ MODERATION CHECKLIST

SUBJECT: INFORMATION TECHNOLOGY			
Teacher's Name:		School:	
Moderator's Name:		School:	
Evidence of attended cluster meetings?			Yes/No
TEACHER'S PORTFOLIO – GENERAL	Programming Language Used?		Delphi/Java

Teacher's portfolio available?	Yes/No	Appendix C?	Yes/No
Appendix D?	Yes/No	Cover sheet with centre's details clearly labelled?	Yes/No
PAT Task sheet available?	Yes/No	PAT Mark sheet available	Yes/No
OOP Test available with the paper analysed to cognitive levels?	Yes/No	OOP Test Memo available	Yes/No
Data Aware Components sheet available?	Yes/No	Data Aware Components sheet available	Yes/No
Preliminary Exam (practical and theory) available with the paper analysed to cognitive levels?	Yes/No	Preliminary Examination marking guidelines (practical and theory) available?	Yes/No
Normalisation Test available with the paper analysed to cognitive levels?	Yes/No	Normalisation Test marking guidelines available?	Yes/No
SQL Test available with the paper analysed to cognitive levels??	Yes/No	SQL Test marking guidelines available?	Yes/No
Theory Test available with the paper analysed to cognitive levels?	Yes/No	Theory Test marking guidelines available?	Yes/No

TEACHER PORTFOLIO - TASKS				
The standard of Performance Assessment Task (PAT)	too easy	easy	appropriate	too difficult
The standard of Data Aware Components Task (SBA)	too easy	easy	appropriate	too difficult
The standard of OOP Test (SBA)	too easy	easy	appropriate	too difficult
The standard of Preliminary Practical exam (SBA)	too easy	easy	appropriate	too difficult
The standard of Preliminary Theory exam (SBA)	too easy	easy	appropriate	too difficult
The standard of the Theory test (SBA)	too easy	easy	appropriate	too difficult
The standard of the SQL Test (SBA)	too easy	easy	appropriate	too difficult

The standard of the Normalisation test	too easy	easy	appropriate	too difficult

LEARNER PORTFOLIOS - MARKING (if learner portfolios required for moderation)			
Marking according to memo?	Yes/No	Allocation of marks justified?	Yes/No

LEARNER PORTFOLIOS – RECORDING (if learner portfolios required for moderation)			
Learner achievement recorded	Yes/No	Appropriate aggregation?	Yes/No

PAT			
Correct use of parameters and subprograms?	Y/N	Correct use of program structure – sequence, selection (if/case) and iteration (loops)?	Y/N
Project divided into Classes/Units	Y/N		
Correct documentation?	Y/N	Project based on a central theme?	Y/N

LEARNER PORTFOLIOS - SBA (if learner portfolios required for moderation)			
Prelim Theory Paper and Practical scripts included?	Y/N	Scripts have been accurately assessed?	Y/N
SQL Test scripts included	Y/N	Scripts have been accurately assessed?	Y/N
Normalisation Test scripts included	Y/N	Scripts have been accurately assessed?	Y/N
Theory Test scripts included	Y/N	Scripts have been accurately assessed?	Y/N
OOP Test script is included?	Y/N	Scripts have been accurately assessed?	Y/N
Data Aware Components Task is included?	Y/N	Scripts have been accurately assessed?	Y/N

Additional Comments:

Describe any interesting/innovative work:

TEACHER'S SIGNATURE:		**DATE:**	
MODRATOR'S SIGNATURE:		**DATE:**	

4. APPENDIX D

NATIONAL SENIOR CERTIFICATE EXAMINATION
INFORMATION TECHNOLOGY
TEACHER'S RECORD OF MARKS

		PAT	**SBA**						
		Performance Assessment Task (PAT)	Data Aware Components Task	OOP Test	SQL Test	Normalisation Test	Theory Test	Prelim	**Total SBA**
Exam No.	**Name**	**100**	**15**	**10**	**10**	**10**	**15**	**40**	**100**

5. APPENDIX E

NATIONAL SENIOR CERTIFICATE EXAMINATION

INFORMATION TECHNOLOGY

SUMMARY OF SBA AND PAT ASSESSMENT

(To be filled in by the candidate and controlled by the teacher. To be included as the 1st Page of the learner's SBA and PAT)

Centre Number: _____ Examination number: _____

Description of task	Brief Description	Application Package	Possible Mark	Actual Marks
Performance Assessment Tasks				
Programming Project			**100**	
		SUBTOTAL	**100**	
School Based Assessment Tasks				
OOP Test			**10**	
Data Aware Components Task			**15**	
SQL Test			**10**	
Normalisation Test			**10**	
Theory Test			**15**	
Prelim			**40**	
		SUBTOTAL	**100**	

DECLARATION BY THE CANDIDATE:

I, _____ (print full names) declare that all the external sources used in my SBA and PAT have been properly referenced and that less than 20% of the code in my Programming Project has been obtained from external sources, as required by the IEB.

Signed: _____ Date: _____

Candidate

DECLARATION BY THE CANDIDATE'S TEACHER:

I _____ (print name and title of teacher) at

_____ (print name of school) declare that the

work provided by this candidate has been monitored and checked for plagiarism.

Signed: _____ Date: _____

Teacher

6. APPENDIX F

NATIONAL SENIOR CERTIFICATE EXAMINATION
INFORMATION TECHNOLOGY
REFERENCING

You will find full details of the Harvard Standard on the Sheffield University site at:
<http://www.shef.ac.uk/library/libdocs/hsl-dvc1.html> and:
<http://www.shef.ac.uk/library/libdocs/hsl-dvc2.html>

A book should be referenced as follows:
Allen, John R. Burke, Michael E. and Johnson, John F. Allen, 1983: *Thinking about Logo.* Holt Rinehart and Winston New York

1. The author's surname, followed by a comma, followed by the first names or initials. If there is more than one author then note all of the authors.
2. The date of the publication.
3. The title in italics.
4. The publisher's name and location.

A webpage should be referenced as follows:
Baldwin, Richard *Java 2D Graphics, Simple Affine Transforms*, 2003. Available from: http://www.developer.com/net/cplus/article.php/626051 [Accessed 17th October 2003]

1. The author's surname, followed by a comma, followed by the first names or initials. If there is more than one author then note all of the authors.
2. The date of the publication.
3. The title in italics.
4. The words 'Available from:' followed by the URL.
5. In square brackets the word 'Accessed' followed by the date when the site was visited.

(Candidates must be aware that the date of their website visits must be noted for the bibliography.)

A list of references
A list, in alphabetical order of author, giving details of the sources quoted in the text, as described above, for books and websites, must appear at the end of the task under the heading 'References'.

Please note the Harvard standard is not the only acceptable way of referencing but it is the one preferred by a number technikons and universities in South Africa. **Whichever method is used, it must be used consistently**.

7. APPENDIX G

NATIONAL SENIOR CERTIFICATE EXAMINATION

INFORMATION TECHNOLOGY

CONTENT TO BE COVERED

System Technologies		
Grade 10	**Grade 11**	**Grade 12**

Basic concepts of computing
 	
• Define Information and Communication Technology
• Overview of a general model of a computer in terms of input, storage, processing, output, communication
• Overview and concepts of the main components of a computer system
  - Hardware vs. software
  - Components of a home computer system: input (keyboard, mouse), storage (hard drive), processing (CPU and RAM), output (monitor, printer), communication (modem/router)
  - System software (operating system) and application software – examples and uses
  - Shareware, freeware, free open source software (FOSS), proprietary software
  - Interdependency of hardware and software
• Types of computers: desktop, notebooks, netbooks, tablets, smartphones, server, embedded computers (microcontrollers): purpose and uses
  - Differentiate between the types of computers in terms of primary uses, processing power and size
  - Categorise computers/classification of computers: Portability/mobility, processing power, usage
• Advantages and disadvantages of using computers
• Explanation of and differentiation between data		

and information - Information processing cycle: input, processing, output, storage, communication - Transition from raw data to processed/organised information - Uses and examples of information within an organisation • What is an ICT system? - Overview of a general model of an ICT system: Convey data, manipulate data, store data - Example of an ICT system (familiar context, e.g. Point-of-Sales system, cell phones)		
Basic concepts of hardware • Define hardware • Input devices - Example: Alternative keyboards, pointing devices, touch screens and touch sensitive pads, pen input, game controllers, digital cameras, video input, scanners and reading devices, data collection devices, biometric input, toy/electronic device interfaces - Transfer/synchronise between computer and phone • Output devices - Display devices, printers, data projectors, interactive whiteboards, toy/electronic device interfaces - Concepts regarding quality of output and speed where applicable • Storage - Hard drives (fixed and portable), USB flash drives, U3 smart drive, solid state drives, memory cards, optical disks, DVD and Blu-ray drives and media - Capacity, portability, use • System unit (processing – CPU and RAM) - General function of CPU, ROM and RAM • Identify ports and connectors and their purpose: USB, Firewire • Categorise hardware according to input, output,	**Hardware** • Describe the motherboard - Purpose and role of the motherboard - Flow/transfer of data between components o Point-to-point serial connections - Components as part of the motherboard o Purpose and role of BIOS chip, CPU, RAM, ROM, Firmware, slots, cards, buses o Purpose and role of cache • Purpose and role of the expansion cards • Memory as part of the computer system - ROM, RAM – role and characteristics - Temporary/permanent/magnetic/optic/solid state • Difference in performance of different components and caching (including Web caching and disk caching) • Plug and play and Hot swapping	**Hardware** • Modular design • Overview of factors influencing performance of a computer - CPU (speed and multi processing) - Memory capacity (cache and RAM) - Storage speed SATA, solid state vs mechanical drives - Network speed, NIC speed, cabling (LAN), wireless vs wired (sharing bandwidth) • Motivate a typical computer system with respect to the hardware needed for a specific purpose computer system for - Home/personal use - Game and entertainment - SOHO (Small-Office-Home-Office) user - Power user • Mobile technologies - Examples: Smart phones, laptops, tablets, netbooks, e-book readers - Advantages of mobility - Constraints · Battery life · Size · Computing power vs. power consumption

storage, processing and communication devices • Memory vs. storage • Compare input, processing, output, storage and communication devices of a desktop computer with a Smartphone.		
Systems Technologies: Basic concepts of system software • Define system software • Operating system - What is an operating system? - What is the purpose/role of an operating system? · General role: Suite/group of related programs which manage hardware and software · Specific role: Provides user Interface, I/O management · Brief overview of the role of the operating system in terms of file, disk, memory, storage and process management - Types of operating systems (also associate with types of computers), e.g. stand-alone, network, embedded - Examples of common operating systems • Utility programs - Define and list uses - Generic/common examples • Purpose of device drivers	**Software** • Overview of various types of operating systems in terms of cost, size, hardware needed, and platform • Define Programming language compilers/interpreters • Overview of processing techniques (managed by systems software) - Multi-tasking, multi-threading, multi-processing - Interrupt – hardware and software • Role and purpose of virtual memory	**Software** • Describe cloud computing - Effect on hardware needs - Software as a service (SaaS) o List advantages and disadvantages o Who owns what? • Overview of virtualisation - Virtual machines – purpose and examples • Virtualisation of servers - List advantages and disadvantages
Computer Management • Describe computer management • Overview and purpose of various management tasks and operating system utilities - Management of desktop - Management of files and folders - General housekeeping tasks - Defragmentation - Scheduling/updating - Archive, backup - Compress/decompress files - Security features – firewall, anti-virus, control of spyware, adware - Installing/uninstalling software (custom and full		**Computer Management** • Factors influencing computer management • Recommend management tasks for general housekeeping and to maintain data integrity and protect the system

installation, product keys, activation codes) - Add devices/drivers - System settings and properties		

Communication Technologies		
Grade 10	**Grade 11**	**Grade 12**
Networks • Describe a network - Reasons for using networks such as communication, access to/sharing resources, centralisation, file and funds transfer, productivity, leisure - Advantages and disadvantages of networks • Overview of different communication media (wired vs. wireless) - Types of cabling - Types of transmitters and components • Classification of networks (PAN, LAN, WAN, GAN) • Local area network (LAN) vs. wide area network (WAN) – coverage and where it is used • General function of communication devices (modem/router) • Differentiate between client-server and peer-to-peer networks • Explain the reasons for logging into a network and connecting to a server – access control	**Networks and e-Communications** • Overview of physical aspects of a network - Data transmission in a LAN o Media (reinforce from Grade 10) o Network Card o Physical layout (topology – star) o Physical limitations (bandwidth) o Connection devices(router/bridge) o Size - WLAN o Wireless access point o Wireless mesh o Wifi and hotspots - Extending a LAN o Fibre optic backbone o Wireless bridge o Routers (all in one router) • WAN - Gateways - Transmission – satellite radio waves and microwave • Wireless technologies - GPS, GSM, GPRS, EDGE, (Enhanced GPRS), 3G, 4G, WiMAX, Wifi, bluetooth, etc. - Difference in range and bandwidth (non-technical) • Mobile technology - mobile browser – description and examples - Compare to PC based browsers • Protocols - How protocols control data, e.g. POP3, SMTP, TCP/IP (UDP vs TCP) - Web protocols - http, https, ftp - Wireless protocols – WAP, HSDPA, HSUPA, USSD	**Networks and e-Communications** • Sharing concepts - Sharing files and folders, user rights, BitTorrent (content distribution protocol that enables efficient software distribution and peer-to-peer sharing of very large file) -Risks and benefits - Online services (e.g. drop box/Mobile Me/Google Drive) • Remote access - On local network, through Internet - Cloud Computing VS VPN
Electronic Communications		

• Describe electronic communication
• Overview of applications/tools to facilitate e-communication – purpose and uses (What is it? What is it used for?)
  - Email, Web browser, File Transfer Protocol (FTP), instant messaging, chat rooms, video conferencing and Voice over Internet Protocol (VoIP), RSS aggregator, Weblog, text, picture and video messaging –Examples
• Email as a form of e-communication
  - Uses of email
  - Email accounts (Internet Service Provider (ISP) and web-based)
  - Email addresses
• Responsible communication styles and netiquette		

Internet Technologies

Grade 10	Grade 11	Grade 12

Internet and WWW
 	
• Describe the Internet
• Internet addresses – Internet protocol (IP) addresses and domain names
• What is needed to connect to the Internet referring to
  - Internet Service Providers (ISPs), wired and wireless connections
• Overview of the World Wide Web (WWW)
  - Describe the WWW
  - Web address/uniform resource locater (URL)
  - Web page and Web site
  - Types of Web sites, their purpose/what they offer and examples
  o Portal, news, informational, business, Weblog (blog), Wiki, online social network, educational, entertainment, advocacy, Web application, content aggregator, personal
• Criteria to evaluate Web sites
  - Affiliation (e.g. who supports the Web site?)
  - Audience (e.g. level at which it is written/who is it intended for?)
  - Authority (e.g. who is the author and what are his/her credentials?)
  - Content (e.g. organisation of content and working links)
  - Currency (e.g. is the information on the Web page up-to-date?)
  - Design (e.g. is it easy to navigate and visually pleasing? How quickly does it download?)
  - Objectivity (e.g. does it reflect any preconceptions?)
• Browsing and searching
  - Examples of Web browsers
  - What is a search engine?
  - Examples of search engines
  - Performing searches using a search engine (search techniques)
  - How to access and browse a Web site
  - What is the World Wide Web consortium (W3C)?
	Internet and WWW
	
	• Overview of multimedia as part of Internet technologies
	  - Download vs. streaming
	  - Live broadcasts
	  - Video on-demand and IPTV (Internet Protocol Television)
	  - VOIP
	• Media
	  - Compression technology (MP3, Mpeg4, Mpeg2, Jpeg) Relate to type of files and their appropriate compression technique.
	  - Compression: Quality vs. bandwidth and speed
	  - Lossy vs Lossless
		Internet and WWW
		
		• Trends and emerging technologies, e.g.
		  - Web 3.0 (Semantic Web)
		  - Web 4.0
		• Improve searching
		  - Semantic search Engines (by context)
		  - Mediated search Engines (by categories)
		  - SEO (Search Engine Optimisation)

• Overview of plug-in applications - Describe plug-in applications - Examples and purpose of plug-in applications for browsers such as PDF converters and tools, Flash player, Java, QuickTime player, RealPlayer, Silverlight		
Internet Services Technologies • Usability of Web pages/sites - Compare usability issues such as readability, navigation, consistency, layout, typography - How does this relate to user interface design? • Overview of Web development - Concept of a Web page as a file that contains text and HTML and/or XHTML code - Tags and elements of a web page - Web design principles • Exploring a simple html editor with basic html code to view tags and elements • Utilising a html preview tool to explore some of the functionality and purpose of the various tags: • <html> <head> <title>Hello HTML</title> </head> <body> Hello World! </body> </html> • As an optional extension, learners could create simple Web pages using HTML/XHTML and a text editor such as Notepad using the following: • Basic Document Tags <html> <head> <title> <body> </html> </head> </title> </body> • Heading Elements • Text Elements • Image • Ordinary Links Link-text 	**Internet Services Technologies** • Overview of the evolution of the Internet in terms of - Software and applications o Static and dynamic sites: Web 1.0, Web 2.0, Web 3.0 - How a web page is accessed in Web 1.0 - Client-side and server-side scripting - Cookies • Internet related careers - Web designer - Web author - Graphics and multimedia designer	**Internet Services Technologies (±½ week / 2 hours)** • Overview of Internet Services technologies - Location based services - Internet sites' accessibility to mobile devices • Overview of supporting technologies: - HTTP, HTTPS, FTP, VoIP, RSS, SEO (search engine optimisation - Rich Internet applications - Security services o public and private key encryption and SSL • Online applications • Storing data - Locally (cookies) - Online (databases) - relate to cloud computing - Role of SQL, scripting languages (e.g. PHP, JavaScript), XML • Running instructions - Locally (scripts, AJAX) - Online (server side, scripts and code) • Internet vs. Intranet vs. Extranet

	Security • Human error (GIGO, accidents) • Accuracy and validity of data - Data input methods o Keyboard vs GUI o Barcode readers o RFID o Online - Invisible (e.g. credit card, loyalty card, government, forms, toll road passes, cellphone) - Data types used - Verification of data - Validation of data e.g. format check, data type check, range check, check digit - Software bugs • Threats - Physical access o Theft o Flash drives and portable media - Hardware failure o Storage o Power - Network vulnerability o Virus, worm, Trojan, rootkit (software enabling administrator access to a computer), spoofing, phishing, WEP/WPA/WPA2 key • Remedies - Backup, UPS, passwords, rights, firewalls, anti-virus, validation, RAID, encryption	

Social Implications
These section can be integrated into the relevant sections above

Grade 10	Grade 11	Grade 12
Social Implications (±1½ week / 6 hours) • Licence agreements (including creative commons), piracy, copyright, copyleft • What are social, ethical and legal issues pertaining to ICTs? • Economic reasons for using computers: Saving paper, labour, communication costs, efficiency, accuracy, reliability • Digital divide - What is the digital divide? - What are digitally enabled citizens? - Reasons for the digital divide • Ergonomics, green computing issues, health issues • Global e-communication, i.e. accuracy, time, distance, communication costs, speed • Email threats and issues: Viruses, hoaxes, spam, phising, email spoofing and pharming • Safe email and Internet use: Dangers and tips to ensure safe use	**Social implications (±2 weeks / 8 hours)** • How the advancement of ICT affects the human race - Computers providing solutions to issues of national and international importance such as o Weather, elections, census • Capabilities and limitations of ICTs • Social engineering, impact of social websites • IT related careers and the effects of digitalisation - Careers: PC technician, programmer, network administrator, graphics design, Web authoring, security consultants, systems analyst - Effect on workplace and employment practices - Mobile offices, virtual office, decentralisation of labour, office automation, robotics, artificial intelligence - Network use policies and practices AUP - Cost benefit analysis of the advantages and disadvantages of a computerised system • Describe the influences of computer and mobile technologies on society due to globalising trends - Online services (online banking, booking reservations, e-learning) - Video conferencing, interactive whiteboards, online banking, cell phone banking, social websites (e.g. Facebook)	**Social Implications (±½ week / 2 hours)** • Reducing the environmental impact of the use of computers • Discuss various ways to stay informed about computer technology • Getting latest product upgrades, viruses and other threads, upgrading • Computer criminals - Hackers, crackers, cyber gangs, virus authors • Types of cyber crimes • Effect of cyber crimes • Computer crimes such as hardware, software, information, identity theft, bandwidth theft, theft of time and services - Internet-related fraud scams - Internet attacks (worms, virus, denial of service, back doors) - Phishing, pharming, siphoning - Unauthorised remote control and administration, e.g. botnets, zombies - Right to access vs. right to privacy, misuse of personal information • Safeguards against computer crimes, threats and criminals • Explain how computers provide solutions to issues of national and international importance such as - Distributed computing power - Decision making • Describe the evolution of social networking and the effect on society - Information overload - Availability of personal information o Consequences of search engines and group communications o Social, political, environmental o Global community – cultural effects o Social websites and social engineering

		○ Wikis • List and discuss issues regarding privacy and information sharing e.g.Google Drive and Drop Box • Cookies, anonymity, Global Unique Identifiers, file sharing – movies, music

Data and Information Management		
Grade 10	**Grade 11**	**Grade 12**

Data representation and storage (±2 weeks / 8 hours)
 	
• Data, information and knowledge
• What is data representation?
• What is data storage?
• Bits and bytes
• Overview of number systems: Decimal, binary, hexadecimal
  - Conversion between
  o binary and decimal and vice versa
  o decimal and hexadecimal and vice versa
• Overview of digital character representation, e.g. ASCII/UTF-8, Unicode
• Overview of primitive data types and their storage (integer types, text/string types, floating point types)
• Overview of data structures and collections of data– storage in terms of
  - Files, databases
  - Reasons for data storage
• Computer file management:
  - Organising files
  - Files, folders and drives
  - File specification: Drive: Path/Filename/File Extension
  - File manager
  - Hierarchical structure
  - Reasons for having a file structure
  - Manipulating files and folders
  - File-naming conventions
  - Common file types and extensions (association)
  o Archived and compressed
  o Forms of text files
  o Database, spreadsheet, presentations and word processing documents
  o Graphic files, movie, sound and animation files
  o Font files
  o Source code
  o Object code, executable files, shared and		

dynamically linked libraries • Saving as another type/version and exporting between file types		
	Database Design and Management (±½ week / 2 hours) • Relationship between data, information, knowledge and decision making • What is a database? • Characteristics of quality data: - Accuracy, correctness, currency, completeness, relevance • Describe database management software (DBMS) • Examples of DBMS software, e.g. - Microsoft SQL Server - Oracle - Microsoft Access - Blackfish - Open source databases, e.g. PostgreSQL, MySQL • Database management – - Size and accessibility - Desktop vs. server (size and accessibility) - Distributed (e.g. Google) vs Centralised • Overview of database-related careers and roles of people involved - DBA (database administrator) - Unix administrators - Programmers - Analysts - Project managers	**Database Management (±½ week / 2 hours)** • Caring for and managing data - Value of data - How to protect data: validation, verification, integrity, logging changes (audit trail), warehousing, controlling access (passwords, security, user rights), parallel data sets • Hacking through data - Invalid/false data - Database management software (DBMS) vulnerability (SQL injection) • Data warehousing - Describe data warehousing - Purpose and uses • Data mining – description and purpose - SQL - Extracting data - Looking for patterns - Discovering knowledge - Strategy - Big data • Location-based data
		Database Design Concepts (2 weeks /8 hours) • Characteristics of a good database - Data integrity - Data independence - Data redundancy - Data security - Data maintenance (ease of) • Problems with databases - Anomalies –Update, Insert and Edit • Key fields - Reinforce primary keys, alternate keys - Foreign keys

		- Composite keys • Design and create a relational database - Normalisation - 1NF, 2NF and 3NF - Data dependencies - Duplicate data, derived data and redundant data • Simple entity relations diagrams (ERD)

Case Studies		
Grade 10	**Grade 11**	**Grade 12**
		Content using Case Studies – All topics (±1½ weeks / 6 hours) Consolidate content, concepts and skills using case studies to: • identify the basic hardware configuration of a computer in terms of the processor, memory and hard drive size • understand computers and their uses • know how to use computers as tools to access information and to communicate with others around the world • make better buying decisions – interpret advertisements and make judgements about quality and usefulness when buying equipment, software • know how to fix simple computer problems and deal with challenges that arise with utilising computers (and know when to call for help) • know what kind of computer uses could benefit and advance work place and career path opportunities • know how to protect themselves against online villains and threats • know how to apply digital tools to communicate, gather, analyse, use information and solve problems • recommend specific hardware/software for a specific scenario

Solution Development		
Grade 10	**Grade 11**	**Grade 12**
•		

Data Structures using Java or Delphi	Data Structures	

• Storage of data using single variables
  - String
  - Real
  - Integer
  - Boolean
• Predefined Objects
  - use of predefined objects
  o instantiate an object
  o constructor methods
  o methods calls to an object)
  o concepts of encapsulation – that an object consists of fields and methods (procedure and functions)
	• **Temporary Storage in RAM**
	  - **Arrays**
	  o Arrays as a data structure for a collection of the same data type (1-dim)
	  o Step through items
	  o parallel arrays
	  - **Objects**
	  o constructor to instantiate an object and assign values to fields
	  o default constructor
	  o destructor methods
	  o overloading constructors
	  o accessor, mutator and toString methods
	  o private and public fields
	  o fields of complex types e.g. objects or arrays of other objects
	  o encapsulation
	  o information hiding
	  o method overloading
	  o parameter passing to send data to a method in an object
	  o return types
	  o Terminology: instances, instantiation
	  o concept of an object as the Back End independent of the User Interface/Front End
	  - **Arrays of Objects**
	  o usually two fields – the array and the number of elements in the array (integer)
	  o constructor to populate including transfer of data from permanent storage to temporary storage
	  o toString to display the array
	  o methods to manipulate such as sorting and searching which are explained in the section on algorithms
	
	• **Permanent Storage**
	  - **Text Files**
	  o files containing a list of fields (one item on a line)	

	o files containing multiple data on each line separated by specific character - **Databases (single table)**	
Program Structures using Java or Delphi • Global variables vs. local variables • Identifier naming conventions **Numbers** • Assigning values to variables • Data types: Integers, strings, reals, Booleans • Keyboard input, mouse input • Operators (+ , - , * , /) and order of precedence • Retrieving remainders: modulus • Comparison operators >, <, =, !=, >= and <= and performing logical comparisons • Boolean logic/operators [and, or, not] • Mathematical functions – random, round, square root, square and power, • Format number output to display a number to a certain number of decimal places **Strings** • String operators such as concatenate/combine strings • String operations such as comparing strings • length or len method • String characters to format output • isolate a character in a string **Conditional constructs** • [if and if-then-else] - including Boolean operators and relational operators - nested if statements • case / switch - using ordinal (integer and char types) - as replacement for nested if statements • Boolean expressions **Iteration constructs** • Infinite vs terminating loops	**Program Structures** **Variables and data types** • Variables in terms of scope and lifetime of variables (global vs local) • Variables and objects as form class attributes • Casting/converting from one type to another (integer/real/float/string/text) **Built in Methods** • Revisit methods for mathematical calculations focussing on parameter passing **String Manipulation** • Basic string operators: concatenation and comparison • String manipulation using string methods: inserting and deleting characters, determine the position of a character, find a character/substring, determine the length of a string • Extend applications by adding auxiliary methods to perform simple string manipulation in the form class/User Interface/Front End or a user defined class/Back End **User Defined Methods** • Relate methods to part of an object rather than a mechanism for structured programming • Void Methods / Procedures • Typed Methods / Functions • Parameter Passing • Helper Methods (private methods in an object) • Accessor and Mutator methods • Static and non static methods (Java) **Parameter Passing / Data Transfer** • Apply parameter passing and return values as communication between the User Interface/Front End (application) and the Object Class / Back End • Use of parameters to send data to methods (single	

- For loops (pretest loop)
  - loop counter ascends and descends
  - loop control variable of integer and char types
  - user determines the number of times the loop is repeated
- While (pre-test loop)
  - SITC to ensure the loop terminates
  - rogue values to terminate the loop
  - terminating on users input
- Do While or Repeat Until loops (post-test loops)
- Compare all three loops and choose the correct loop when coding
- Nested loops

Debugging techniques
- Debugging using the variable watch facility
- Simple diagramming and animation
- Simple condition based drawing

type, object, array, array of objects)
- Typed methods / functions to return data from methods
- Data returned as a single variable, object, string with fields separated by a # or similar character, array or array of objects
- Parameters as a mechanism to reduce code and make methods more generic

Exception Handling
- Basic Exception handling to deal with a missing fileInput validation using code constructs

Graphical User Environment Builder (some can be covered in Grade 10 if a high level programming language is used)
- Controls/components
  - Create, retrieve and save an application/project for development and modification
  - Controls/components to be used in this section:
  - Form, button, label, panel, radio group, text box, combo box, check box, message box, image, timer
  - Naming conventions
- Form class / User Interface/ Front End
  - Objects and attributes/properties
  - Common properties of components
  - Events handlers, methods (code to handle events)
- Adding controls to a form and enter simple code (simple applications using basic controls that accept input and give output such as simple messages)
- Manipulate and access simple component properties
- Write simple code to display a message, e.g. to display simple messages using a message box
- Illustrate the concept of inputting data in the User Interface using a GUI component and passing the data as a parameter to User Defined Object/ Back End (the idea of passing a message to an object)
- Passing data from the Back End methods to the Front End/User Interface using return types of methods.
- Add controls and methods to perform tasks such as closing a form, changing colours, administer a login,

	manipulating the appearance and behaviour of other controls • Simple Graphical User Interface(GUI) design - concepts, e.g. functionality and usability issues **Debugging** • Errors and debugging techniques • Implement watches and traces to identify logical errors in code constructs (compiler based and paper based)	
Algorithms (±3 weeks / 12 hours) using Java or Delphi • Basic concepts of an algorithm • Examples of algorithms in everyday life, e.g. instructions to draw a kite or fold a paper jet, recipe to bake a cake, perform long division • Devise an algorithm/basic instructions to complete similar tasks • Use a tool, e.g. basic flowchart to describe a task • Interpret a basic flow chart • Produce an algorithm to solve a problem • Tools, e.g. basic flow charts/pseudo code to represent an algorithm • Trace an algorithm to determine the outcome or the correctness – trace table • Value of accurate, well-tested algorithms • Basic calculations such as area, volume, VAT and simple formulae, typical calculations done in other subjects • Determine smallest, largest value of more than two values • Swapping values • Determining aggregates e.g. sum • Determine whether a number is even • Determine whether a number is a factor of another number • Isolate digits in an integer number • Determine if a number is a factor of another number, • Basic validation techniques (input and processing),	**Algorithms** • Implement algorithms to solve general computing problems in similar categories to those given below: - Finding the smallest/biggest of more than two numbers - Determine whether a number is a prime number - Lowest common multiple (LCM), greatest common divisor (GCD) - Determine current age given birth date **Validation** • Basic input and processing validation techniques, e.g. test for division by zero • Use exception handling to validate data **String Manipulation** - Use ID number to determine age, gender - Isolate initials from a surname i.e. Change Fred John Smith to FJ Smith - Isolate a word in a sentence with punctuation - Encode / encrypt a string using a simple algorithm - Remove vowels from a string **Arrays** • Basic operations, e.g. aggregates, highest, lowest • Insert and delete an element (include shift up and shift down) - Delete duplicates • Sorting - Compare selection, bubble, improved bubble and insertion sort	**Algorithms** • Use programming language constructs in the execution of various simple database transactions - Access fields and records within a dataset with code constructs and applicable methods - Navigate the records of a dataset - Modify individual fields and records within a dataset with code constructs and applicable methods, and apply all changes - Manipulate a dataset object and records with code constructs and apply all changes - Incorporate dataset event handlers and methods as part of the solution - Use common dataset event handlers in the development of a solution • Query a database using a join on a maximum of two tables with multiple criteria (the database may contain more than two tables, however a maximum of two tables is joined for query purposes) – reinforce • Develop a multi-form GUI incorporating simple controls • Share data amongst forms as part of the solution • Use appropriate algorithms and/or built in methods in the manipulation of data such as sorting routines, string based routines, date and time • Incorporate defensive programming techniques as part of the solution (data validation) - Check for empty fields, ranges, valid formats, data • Design and develop code constructs to generate a text based report

e.g. test for negative number when calculating square root
• Find a specified character in a string
	• Searching using the linear search algorithm
	  - Compare linear search and binary search algorithm
	
	Text Files
	• Read from a file
	• Write to a file (include rewrite and append)
	• Generating a simple text-based report, e.g. summary of data
	• Apply simple file input and output using a text file to populate data structures and to develop simple reports
	
	Databases
	• Grouping data
	  - Records and fields
	  - Tables
	• Create a simple database with focus on table design without relationships
	• Data maintenance tasks such as
	  - Insert/add, delete, edit
	  - Process, sort, query (generating information from a database)
	• SQL to access a single table database
	  - SELECT, DISTINCT
	  - INSERT, UPDATE, DELETE
	  - WHERE
	  - ORDER BY including LIMIT / TOP
	  - GROUP BY
	  - Special operators: BETWEEN , IN, LIKE, IS NULL
	  - Creating calculated fields, concatenating fields, reaming fields using AS
	  - Formatting with ROUND, INT, etc.
	  - Casting a field
	  - Mathematical operators including MOD and DIV / integer division
	  - Aggregate functions: SUM, AVERAGE, MIN, MAX, COUNT
	  - Common date functions NOW, YEAR, MONTH and DAY
	  - String functions (LENGTH, MID, LEFT, RIGHT)
		• Construct more complex algorithms, e.g. remove duplicates from a table, extract a random row from a table
		• Develop solutions for various problems using computational thinking and applying software engineering principles that include both database and non-database problems
		  - Test and validate a solution against a set of design specifications
		  - Alter a solution to meet a set of design specifications
		  - Document a solution design and development
		• Motivate the design and development of the solution
		• Evaluate a solution against other solutions
		
		Databases
		• Simple sub queries (nested queries)
		• SQL to access normalised database
		  - INNER JOIN and LEFT JOIN on a maximum of three tables (database may have more than two tables)

	• Accessing a database through programming language constructs • Setup a connection or connect to a database (single table) by providing path in code statements • Query and edit a database (single table) using simple SQL constructs SELECT, UPATE, INSERT and DELETE	
		Application Development using a high-level programming language (±4 weeks / 10 hours) • Problems focusing on problem solving, OOP constructs and principles and algorithmic thinking • Select an appropriate data structure (single variables, arrays, object, array of objects) to model a problem domain and explain the implication of the data structure o compare data structures and algorithms such as arrays to single variables in algorithms such as determining the average, lowest and highest of a set of numbers o Compare arrays of objects to parallel arrays • Comparing algorithms to built-in methods for performing same/similar tasks • Use algorithmic thinking and software engineering principles to develop solutions for a variety of problems, focusing on computational problems which could include a database as part of the solution: - Apply generic algorithms as part of the solution - Devise a specific algorithm(s) where applicable to solve a problem utilising user-defined code constructs or built-in methods - Contrast generic algorithms to built-in methods - Adapt generic methods to solve a similar problem • Validate the solution against a set of data using different techniques, e.g. trace tables, watches, manual output comparison • Alter a solution to meet a set of design specifications • Document a solution design and development PAT • Motivate the design and development of the solution • Evaluate a solution against other solutions

Solution Development: Software Engineering Principles (±1½ week / 6 hours)

- What is problem solving?
  - Understand the problem (task/problem description or scenario/user stories)
    - State in own words
    - Clarity on what needs to be done
    - What is known or given?
    - What is missing or needed?
    - Devise a plan/algorithm
    - Look for patterns
    - Look at related problems, known solutions
    - Examine simpler or special cases
    - Make a table, create diagram, use guess and check, work backwards, identify sub-goal
  - Carry out the plan/implement the algorithm (write the code)
  - Look back/test (see if it works)
- Solve a problem using the problem solving steps including IPO (input Processing Output)
- Acceptance tests (does the program meet the requirements?)

Solution Development: Software Engineering Principles and PAT (±2 weeks / 4 hours)

- What is software development?
- Planning and implementing a solution
  - Define/understand the problem/task
    - Read the specifications and analyse the problem/task to determine the requirements
  - Design the interface and the solution
    - Develop a logical solution based on the specifications and analysis as well as sound software engineering principles
      - Consider functionality and usability issues in designing the interface
  - Code/implement
    - Incorporate suitable programming constructs in the development of a solution
  - Test and debug the program
    - Use testing and debugging techniques and methods
  - Document, implement and maintain the program
- Planning techniques using any appropriate tools

Notes

Diagrams/visual tools for design purposes:

- Use any appropriate tools/techniques:
- TOE (Task, Objects, Events) charts
- Noun-Verb analysis
- IPO diagrams
- UML:
- Use case diagram
- Overview and comparison of different methodologies such as waterfall, rapid application development (RAD), incremental, agile
- Start with PAT – Task description and analysis of requirements using an appropriate methodology
- Reinforce software engineering principles
- Interface design: Functionality and usability principles and program design
- Practical Assessment Task – Design objects / classes
- Design secondary data storage structure (database or text files)

		• Design algorithms to solve the problem using generic / specific / built-in / adapted or user defined • Code solution • Test • Document