FURTHER EDUCATION & TRAINING PHASE (FET)

INFORMATION TECHNOLOGY

SBA EXEMPLAR BOOKLET

GRADES 10-12

basic education
Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

Read to Lead
A Reading Nation is a Leading Nation

2030 NDP

FOREWORD

The Department of Basic Education has pleasure in releasing a subject exemplar booklet for School Based Assessment (SBA) to assist and guide teachers with the setting and development of standardised SBA tasks and assessment tools. The SBA booklets have been written by teams of subject specialists to assist teachers to adapt teaching and learning methods to improve learner performance and the quality and management of SBA.

The primary purpose of this SBA exemplar booklet is to improve the quality of teaching and assessment (both formal and informal) as well as the learner's process of learning and understanding of the subject content. Assessment of and for learning is an ongoing process that develops from the interaction of teaching, learning and assessment. To improve learner performance, assessment needs to support and drive focused, effective teaching.

School Based Assessment forms an integral part of teaching and learning, its value as a yardstick of effective quality learning and teaching is firmly recognised. Through assessment, the needs of the learner are not only diagnosed for remediation, but it also assists to improve the quality of teaching and learning. The information provided through quality assessment is therefore valuable for teacher planning as part of improving learning outcomes.

Assessment tasks should be designed with care to cover the prescribed content and skills of the subject as well as include the correct range of cognitive demand and levels of difficulty. For fair assessment practice, the teacher must ensure that the learner understands the content and has been exposed to extensive informal assessment opportunities before doing a formal assessment activity.

The exemplar tasks contained in this booklet, developed to the best standard in the subject, is aimed to illustrate best practices in terms of setting formal and informal assessment. Teachers are encouraged to use the exemplar tasks as models to set their own formal and informal assessment activities.

MR HM MWELI
DIRECTOR-GENERAL
DATE: 13/09/2017

Table of Contents

1. Introduction

Assessment is the process of evaluating a learner's attainment of knowledge, understanding and skills.

> *There is no decision that teachers make that has a greater impact on learners' opportunities to learn and on their perceptions about what a subject is than the selection or creation of tasks*

School-based assessment (SBA) is conducted by the teacher at the school level and is summative, i.e. it assesses performance against curriculum objectives.

SBA may take place at different points of the learning process, as described through Section 4 in the Curriculum and Assessment Policy Statement (CAPS). Assessment results are recorded and count towards a learner's final promotion or certification.

However, assessment should always contribute to a learner's learning and progress. SBA, therefore, also provides information on a learner's attainment of knowledge, understanding and skills and is used to contribute to individual learning by reinforcing and complementing that learning.

2. Aims and objectives

- Provide quality-assured examples of assessment tasks to capacitate teachers in the setting of SBA tasks.
- Provide guidance to teachers when setting SBA tasks.
- Deepen understanding of the cognitive demand of a task.

3. Assessment Tasks

The assessment tasks included are for Grades 10 -11 and make provision for practical tests. These tests are a collection of assessment methods and questions that samples a domain of knowledge and/or skills.

In IT, the practical component involves algorithms (a portion of program code or a code pattern designed to achieve a task within a program. From an object-oriented perspective, a design pattern would be the equivalent of an algorithm) and processes (the procedure that a person might learn or create to be able to write a code segment). Examples of processes are code tracing, desk checking, translation from design to code, and implementing a known algorithm/structure, which are regarded as process knowledge and are tested through practical tests and examinations.

4. Programme of Assessment (PoA)

IT uses mostly tests and examinations (questioning) to assess knowledge and skills. Questions could include practical tasks and case studies (description of an event, usually in the form of a piece of text, a picture or an electronic recording that concerns a realistic situation) where learners are prompted to analyse the situation, draw conclusions /make decisions/ suggest courses of action.

The PoA also includes a project. The Practical Assessment Task (PAT) generates evidence through evaluation of the software development process and the software development product that includes research/investigation, analysis, design and implementation.

The PAT further provides evidence for a range of knowledge, skills and understanding within and across more than one topic. It therefore benefits learning and helps to make the assessment process more meaningful for learners and give assurance of overall competence.

5. Quality Assurance Process

Quality assurance of SBA is the planned and systematic process of ensuring that SBA tasks are valid, reliable, practicable, as well as equitable and fair and thus increasing public confidence in SBA. This would include all the activities that take place before, during and after the actual assessment, that contribute to an improved quality of SBA.

This booklet focuses mainly on the process of setting quality SBA tasks.

Setting of tasks

Guidelines towards setting quality SBA tasks:

- Know the curriculum and its requirements to identify the knowledge, understanding and skills which are to be assessed.
- Ensure that the assessment allows learners to show that they have the required knowledge, understanding and skills to meet the national standards.
- Ensure that the scenarios or contexts are open and comprehensible to all learners.
- Ensure that the appropriate reading level is used. Tools to determine the reading level of a document are available in most word-processing software.
- Ensure that no part of the assessment has an adverse impact on specific groups of learners, e.g. disabled learners.
- Ensure that all illustrative material reflect an inclusive view of society and promotes equality.
- Consider time.

Construction features to consider when setting tests and examinations:

- The language used in the question paper should not be a barrier.
- The weighting given to a particular part of the question paper reflects its relative importance.
- Sampling is systematic but unpredictable to avoid question ‘spotting’.
- The cognitive demand of the paper is appropriate, i.e. includes lower-order, middle-order and higher-order demands to the prescribed ratio.
- The level of difficulty of the individual questions is appropriate and the level of difficulty of the overall paper is appropriate to the level of the grade.
- The mark available for each question matches the demands of the task and the test specification.
- The memorandum allows for a range of valid answers, especially for open-ended questions.
- Different types of questions are used.

Quality assurance helps to support teachers and build expertise and capacity in the education system to deliver positive outcomes for learners. Through sharing, understanding and applying standards and expectations, quality assurance helps to raise standards and expectations and levels of consistency across teachers and schools.

Moderation of tasks

Moderation is the term used to describe approaches for arriving at a shared understanding of standards and expectations. It further helps to ensure that there is an appropriate focus on outcomes for learners, that learning is at the appropriate level and that learners develop the skills for learning, including higher-order thinking skills, which will allow them to be successful in the future.

Moderation of SBA tasks, **prior** to the administration of the assessment tasks, involves teachers and other professionals, such as specialist senior teachers, heads of department or subject advisors, as appropriate, working together, drawing on guidance and exemplification and building on standards and expectations to check that SBA tasks provide learners with fair and valid opportunities to meet the standards and expectations **before** assessments are used.

Moderation of the assessment task should be done using the following evaluation criteria:

- the assessment tasks are aligned to the CAPS;
- assessments tasks and tools are valid, fair, and practicable;
- the instructions relating to the assessment tasks are clearly stated;
- the content must be in keeping with what the learner has been exposed to;
- the assessment task must be free of any bias;
- the language of the assessment task is in keeping with the language level of the learners that it is designed for; and
- the cognitive and difficulty levels at which the assessment tasks are pitched are consistent with the requirements as stipulated in the CAPS.

Teachers involved in developing their assessment approaches through participation in moderation activities is a highly effective form of professional development.

Further moderation activities will generally take place **after** the assessment task is administered.

6. Cognitive and difficulty levels in IT

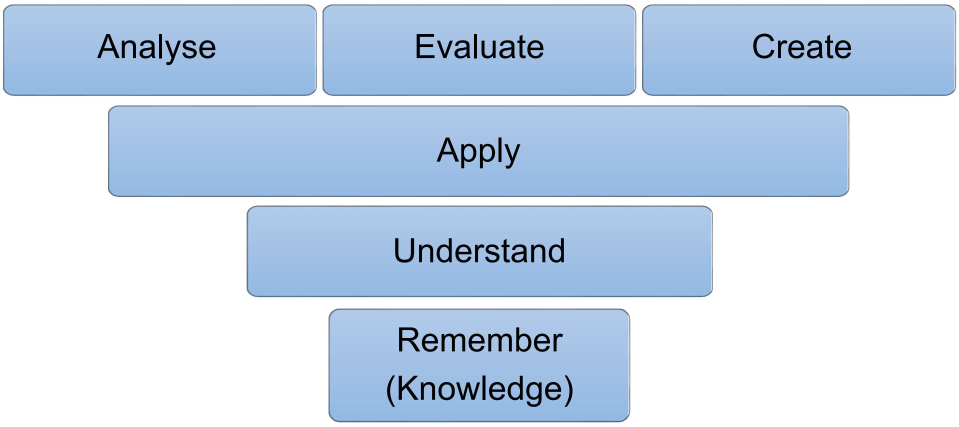
> *All questions are not created equal – different questions require different levels and kinds of learner thinking.*

Cognitive Levels

The cognitive demand of a question is the kind and level of thinking required of learners to successfully engage with and answer a question.

- High cognitive questions are those which demand that the learners manipulate bits of information previously learned to create and support an answer with logically reasoned evidence. This sort of question is usually interpretive, evaluative, inquiry-based, inferential, synthesis-based and open-ended.
- Lower cognitive questions are more basic. They ask learners to recall material previously presented and learned. No or very little thinking and reasoning is required. These questions are generally direct, closed, recall-related and questions that measure knowledge only – factual and process.

Bloom’s revised taxonomy illustrates the different cognitive levels:



Analyse Evaluate Create

Apply

Understand

Remember (Knowledge)

Interpretation of cognitive levels in IT

Cognitive Level		Comment	Context
C1	**Knowledge** Recall of factual/process knowledge ***in isolation***, i.e. one step/set of steps/ instruction/ process at a time.	Requires recalling or recognising only. Practised or learned the isolatable bit, e.g. fact/skill/ process/steps before.	***Exactly*** the ***same context*** as a textbook example or a classroom-based exercise. Explicitly part of the curriculum.
C2	**Understanding** Convert from one form of representation to another.	Requires knowledge and understanding of steps/process/ isolatable bits. Translating ‘words’, pictures, symbols, diagrams into e.g. programming code.	**Familiar context** Includes interpreting, exemplifying, classifying, summarising, inferring, comparing, and explaining.
C3	**Application** Using ***known routines/ steps/ processes*** to complete a task. ***All the information required is immediately available to the learner.***	Requires knowledge, understanding and use of **steps/ *routines/ processes.*** Application of appropriate abstraction **without having to be prompted**…and **without having to be shown how to use it in a familiar context.**	***Familiar context*** but with ***new*** elements / ***new*** circumstances. Learners have seen the same or very similar steps working with different data or other circumstances.
C4	**Analysis** Understand how parts relate to a whole (pinpoint the core/main aspects) or interact with each other and **use appropriate** methods to complete task/solve problem.	Requires reasoning/ investigation/ developing a plan or algorithm; has some complexity. Completing task could have more than one possible approach. Organising component parts to achieve an overall objective.	**New context** Unseen, **unfamiliar** problems/ tasks.
C5	**Evaluation** Judging or deciding according to some set of criteria, generally without real right or wrong answers.	Requires weighing possibilities, deciding on most appropriate. Testing to locate errors.	
C6	**Create** Putting elements together to form a coherent or functional whole; or re-organising elements into a new pattern or structure.	Requires familiarisation with the task by exploring different approaches, interpreting and analysing relevant approaches. Generalisation.	**Novel** situation. The learner has ***no familiarity with completed functional whole***.

In IT, the above cognitive levels are collapsed providing for lower-order, middle-order and higher-order levels:

Lower Order **C1**	**Middle Order** **C2**	**Higher Order** **C3**
Recall (Knowledge)	Understand & Analyse	Analyse, Evaluate & Create
30%	40%	30%

Difficulty Levels

The difficulty level of a question refers to the ease with which a learner can answer a question. It is described as easy, moderately difficult, difficult or very difficult.

What makes a question difficult?

One or more of the following influences the difficulty level of a question:

- Content (subject/concept/facts/principles/procedures), e.g.
    - Advanced content is generally more difficult or content learned in Grade 10 and that is repeated and practised in Grade 11 and 12 usually becomes easier by Grade 12.
    - Number of steps required or the length of the answer could influence difficulty.
- Stimulus (item/question)
    - Language, text or scenario used could influence difficulty.
    - Re-read required or limited time could influence difficulty.
- Task (process)
    - Short questions vs. paragraph or essay – answers that require extended writing are generally more difficult.
    - Steps provided or scaffolding of questions – open-ended questions are generally more difficult than structured questions, i.e. questions that lead or guide learners.
- Expected Response
    - Mark scheme, memo, e.g. detail required in memo vs. detail expected in question.
    - Allocation of marks.

Note: *Within each cognitive level, there exist different difficulty levels.*

Exemplar SBA Tasks and Memos

By determining the cognitive demands of tasks and being cognisant of the features of tasks that make them high-level or low-level tasks, one will be able to select or modify tasks that allow opportunities for all learners.

The level and kind of thinking in which learners engage determine what they will learn.

7.1 Grade 10 Term 1 - Practical Test

Task Description	Comments
This following task is a Grade 10 practical test that covers the practical content and skills for term 1. The test covers • General GUI components • General programming concepts and calculations	At this stage, IT is a new subject with new learners that have no foundation on which to build. This has an impact on how assessment in term 1 is done. As this is the first practical test for Grade 10 IT learners, it is difficult to achieve the correct 30:40:30 ratio between the cognitive levels. The questions are scaffolded and tasks are mostly broken down to make it easier and to enable a learner to code parts in isolation.

Cognitive Analysis

Cognitive Level	Lower Order	Middle Order	Higher Order	Total
Q1				
Q1.1.1	2	2		**4**
Q1.1.2	4			**4**
Q1.1.3		3		**3**
Q1.2.1		5		**5**
Q1.2.2	3			**3**
Q2				
Q2.1	2			**2**
Q2.2	2			**2**
Q2.3.1	3	2		**5**
Q2.3.2		4		**4**
Q2.3.3		4		**4**
Q3	5		10	**15**
Total	**21 (42%)**	**20 (39%)**	**10 (19%)**	**51**

Grade 10 Term 1	Marks:	[51	Time:	75 minutes

Instructions:

- You are required to answer ALL the questions.
- Use meaningful variable names that follow the correct naming conventions.
- Add comments if necessary to explain your code.
- Save all you work in the provided exam drive.
- Rename the gr. 11 data files folder by adding your name at the end.

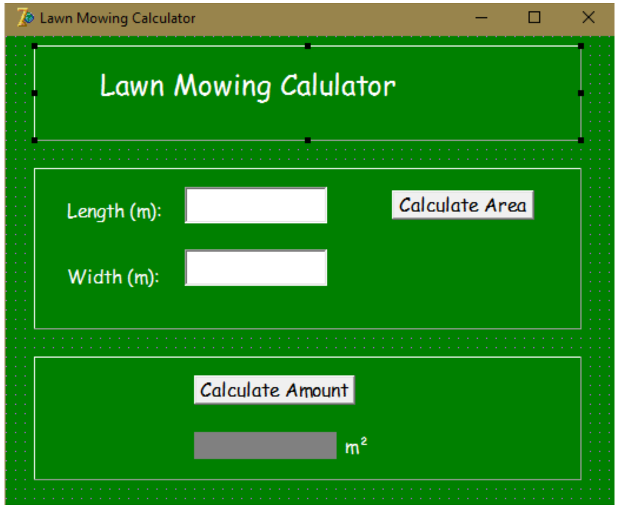
Scenario:

You are planning to go on the provincial hockey tour to Harare, Zimbabwe, during the July holidays and your parents insisted that you earn and save some pocket money to use on the tour. You decided to create 3 apps to help you keep track of the money. The first two apps calculate the money you make, and the third app is used to convert the money to dollars (because Zimbabwe uses US Dollars as their currency) to see how much spending money you will have.

Question 1	Lawn mowing calculator	[19]	Cognitive Demand/Comments

To earn some money, you mow lawns for your neighbours and family. You charge a rate per m^2. Open the Delphi program ***frmMowLawn_p.dproj*** in the Question1 folder. The form contains the following GUI components:

- ○ pnlTop
  - ○ lblHeading
- ○ pnlInput
  - ○ edtLenght
  - ○ edtWidth
  - ○ btnCalcArea
  - ○ lblArea
- ○ pnlOutput
  - ○ btnCalcAmount
  - ○ lblAmount

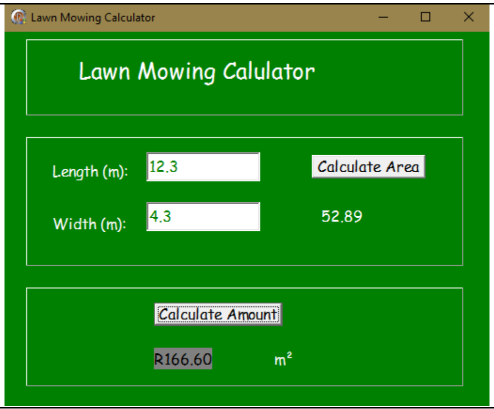


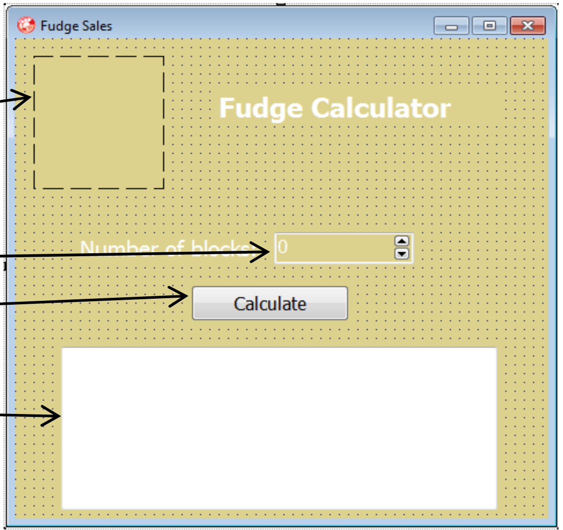
When the form is first opened, pnlOutput is disabled (and therefore all the components on it are disabled too).

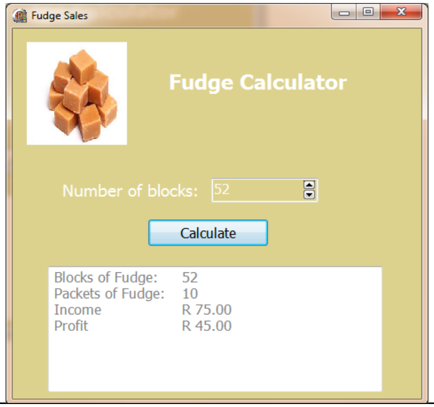
Example of structured question – consists of a stem (which describes a situation followed by a series of related questions). It leads learners through the question and they are less likely to stray from the subject or miss the point.

Explicit, direct instructions – mostly no interpretation required – everything needed to answer the questions is immediately available to the learner. Learners' tasks are clearly indicated.

Mostly, coding for each instruction can be done in isolation, however, care must be taken that failure in one part does not affect the learners' answers in subsequent parts.

1.1	Create an ***onClick*** event to ***btnCalcArea***.		
1.1.1	Declare the necessary variables (a global variable ***rArea*** has been declared for your use already). - rLength - rWidth	**4**	*Knowledge & Understanding* Declaring the variables. Declare the correct data type
1.1.2	Get the length and width as input from the two text edits in ***pnlInput*** and assign the values to your variables.	**4**	*Knowledge* *Exactly the same context* as classroom-based exercise
1.1.3	Calculate the area of the lawn by using the following algorithm: - Calculate area (area ← length * width) - Display area	**3**	*Understanding (mostly)* Translating an algorithm from one form of representation to another form (code)
1.2	Create an ***onClick*** event to ***btnCalcAmount***		
1.2.1	Calculate the amount of money earned. Your fees are R3.15 per m^2 area. Declare this fee as a constant.	**5**	*Understanding & Application* Some thinking and application of concepts required. Learner needs to interpret GUI and apply knowledge to calculate amount.
1.2.2	Display the output in lblAmount as currency. Use the values as displayed in the example to test your program.  Lawn Mowing Calculator Lawn Mowing Calulator Calculate Area Calculate Amount m^2 Length (m): 12.3 Width (m): 4.3 52.89 R166.60	**3**	*Knowledge* The output is clearly provided and learners have practised this before as part of a classroom-based exercise and a clear indication of how this must be done is provided as well as an example.

Question 2	**Fudge Sales**	**[17]**	**Cognitive Demand/Comments**
During the week, you make fudge and sell it at school. It costs you R30 to make a batch of fudge that gives approximately 50 blocks of fudge. You sell the fudge in packets of 5 blocks each, for R7.50 a packet. Any blocks that break during the preparation and packaging of the fudge are not sold. Open the Delphi program ***frmFudge_p.dproj*** in the Question2 folder. The form contains the following GUI components: • imgFudge • spnBlocks • btnCalculate • memOut  Fudge Sales Fudge Calculator Number of blocks 0 Calculate			Learner data:
2.1	Insert the picture Fudge.jpg into ***imgFudge*** and resize it to fit in the block.	**2**	*Knowledge*
2.2	Set the minimum value of ***SpnBlocks*** to 40 and the maximum to 55 so that the spinner can just operate between 40 and 55.	**2**	*Knowledge* Process knowledge
2.3.1	Get the number of usable blocks of fudge from the spinner and calculate the number of packets that can be filled.	**5**	*Knowledge and understanding* Obtain blocks – knowledge Number of packets – *understanding* (round down)
2.3.2	Calculate the income and the profit made from selling the packets of fudge.	**4**	*Application*

2.3.3	Display the output as follows in ***memOut***: (your values will differ depending on your input).  Number of blocks: 52 Blocks of Fudge: 52 Packets of Fudge: 10 Income R 75.00 Profit R 45.00	**4**	*Understanding* Interpretation from screenshot

Question 3	**Money converter**	**[15]**	**Cognitive Demand/Comments**
The final app can be used to calculate the amount of dollars you spend. $1.00 = R13.03			*Understanding and Analysis* Open-ended question that requires interpretation, planning and design.

Memorandum

Question 1

1.1.1	procedure TfrmMowLawn.btnCalcAreaClick (Sender: TObject);
	var
	rLength, rWidth: real;
		Button Click event✓
		Declare one variable ✓
		as the correct data type✓
		Same skills applied to second variable ✓
			4	

1.1.2	begin
	rLength := StrToFloat(edtLength.Text);
	rWidth := StrToFloat(edtWidth.Text);
		Get length & width as input from the edits:
		Get length ✓
		Convert to float ✓
		Correct text box used ✓
		Same skills applied to width ✓
			4	

1.1.3	rArea := rLength * rWidth;
	lblArea.Caption := FloatToStr(rArea);
		calculate area ✓
		convert to string✓
		display ✓
			3	

1.2.1	procedure TfrmMowLawn.btnCalcAmountClick (Sender: TObject);
	const cFee = 3.15
	var
	rAmount: real;
	begin
	rAmount := rRate * rFee;
		Declare constant ✓
		rAmount ✓ = rArea ✓ *✓ cFee ✓
			5	No marks are given for button click event and declaring variables as it was already tested in a previous question.

1.2.2	lblAmount.caption := FloatToStrF(rAmount, ffCurrency, 6, 2));
	end;
		Assigned to the label ✓
		FloatToStrF✓
		Amount earned formatted as currency✓
			3	

Question 2

2.1	Image inserted ✓ Stretched to fit ✓		**2**	Check in properties
2.2	Spinner min = 40✓ Spinner max = 55✓		**2**	Check in properties

2.3.1	procedure TfrmFudge.btnCalculateClick
	(Sender: TObject);
	var
	iBlocks, iPackets : integer;
	rIncome, rProfit : real;
	begin
	iBlocks := spnBlocks.Value;
	iPackets := iBlocks DIV 5; // OR Floor(iBlocks/5);
		Declare blocks & packets as integers✓
		Declare money as real✓
		Get blocks from spinner✓
		Calculate no. of packets: blocks/5✓
		Trunk/floor/DIV✓
			5	

2.3.2	rIncome := iPackets * 7.50;
	rProfit := rIncome - 30;
		Calculate:
		income (use correct variable✓ and value✓)
		profit (use correct variable✓ and value✓)
			4	

2.3.3	with memOut.Lines do
	begin
	add('Blocks of Fudge:' + #9 + IntToStr(iBlocks));
	add('Packets of Fudge:' + #9 + IntToStr(iPackets));
	add('Income' + #9 + #9 +
	FloatToStrF(rIncome,ffCurrency,6,2));
	add('Profit' + #9 + #9 +
	FloatToStrF(rProfit,ffCurrency,6,2));
	end;
	end;
		Display in memo:
		labels✓
		values✓
		money formatted as currency ✓
		lined up in columns using tab ✓
		correct alignment ✓
			5	Lined up in columns requires some basic analysis (easy). Learners have to look at the output and align it in columns.

Question 3

3.1	GUI design Form renamed & saved as frmExchange✓ button✓ input component or inputbox✓ output component✓ all components suitably renamed✓		**5**	Knowledge – Learners can create the GUI from previous experience

	procedure TfrmExchange.btnConvertClick(Sender:
	TObject);
	var
	rRandAmount, rDollarAmount : real;
	const
	// OR rRandAmount / 14.75;
	end;
	EXCHANGERATE = 14.75;
	begin
	rRandAmount := StrToFloat(edtRandAmount.text);
	rDollarAmount := rRandAmount / EXCHANGERATE;
		Create variables✓
		Use 14.75 as exchange rate✓ (does **not** have to be a constant or even a variable)
		Get RandAmount as input✓ (edit, inputbox, spinner), ensuring numerical format✓
		Calculate dollars✓✓
			6	Analysis – Learners need to solve the problem without guidance.

	with memOutput.lines do
	begin
	add('Amount in Rand: R ' +
	FloatToStrF(rRandAmount,ffFixed,6,2));
	add('Amount in Euro: $ ' +
	FloatToStrF(rDollarAmount,ffFixed,6,2));
	end;
		Display:
		labels✓
		unit symbols✓
		formatted ✓to 2 decimals✓
			4	

7.2 Grade 10 Term 2 - Practical Test

Task Description	**Comments**
This following task is a Grade 10 practical test that covers the practical content and skills for term 1. The test covers • General GUI components • General programming concepts and calculations	

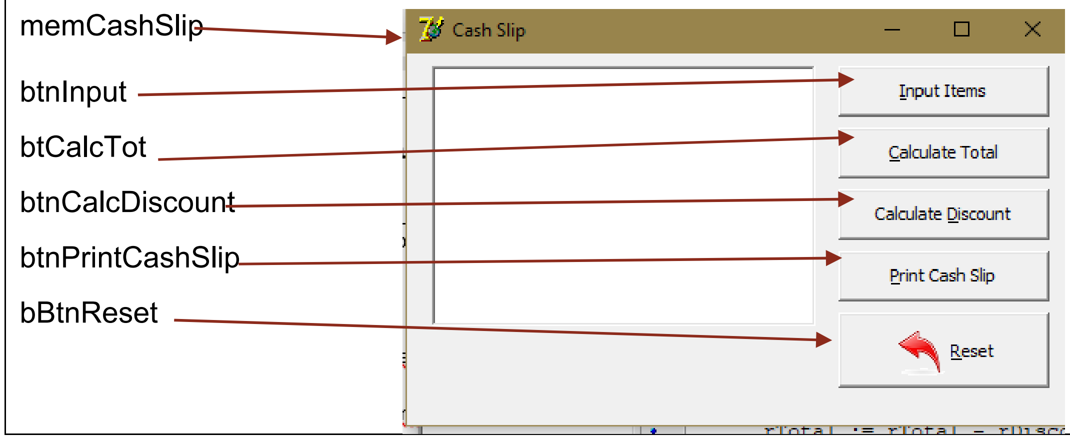
Cognitive Analysis

Cognitive Level	**Lower Order**	**Middle Order**	**Higher Order**	**Total**
Q1.1.1	4			**4**
Q1.2.1	4			**4**
Q1.2.2	1			**1**
Q1.2.3	4			**4**
Q1.3.1	2			**2**
Q1.3.2	2			**2**
Q1.4		8		**8**
Q1.5.1	1			**1**
Q2.1		20		**20**
Q3			19	**19**
Total	**18 (27,7%)**	**28 (43,1%)**	**19 (29,2%)**	**65**

Grade 10 Term 2	**Marks:**	**[65]**	**Time:**	**65 minutes**

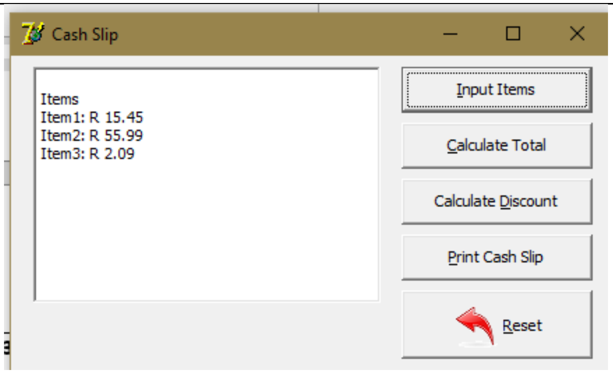
Instructions:

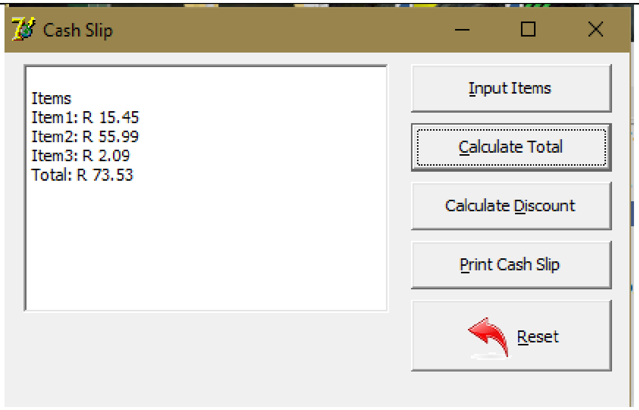
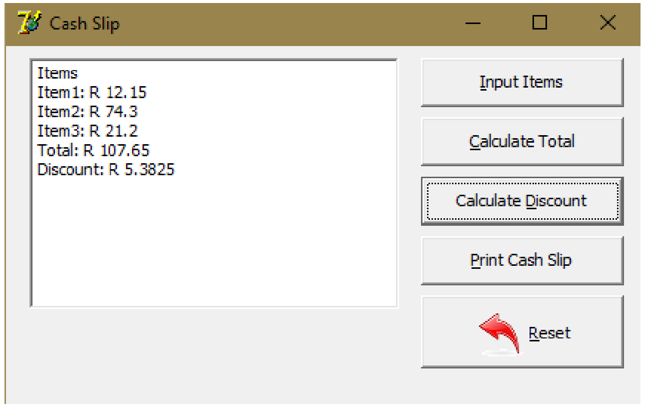
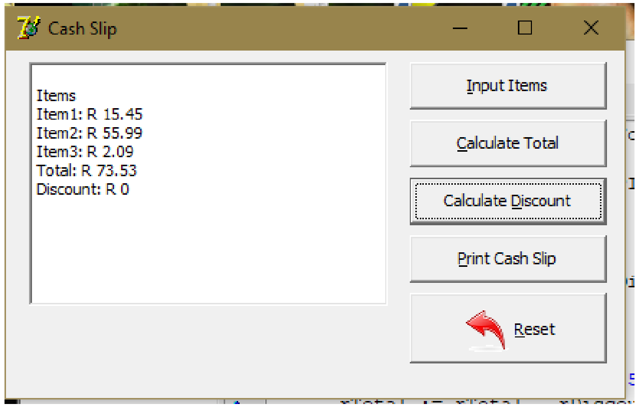
- You are required to answer ALL the questions.
- Use meaningful variable names that follow the correct naming conventions.
- Add comments if necessary to explain your code.
- Save all you work in the provided exam drive.
- Rename the gr. 10 data files folder by adding your name at the end.

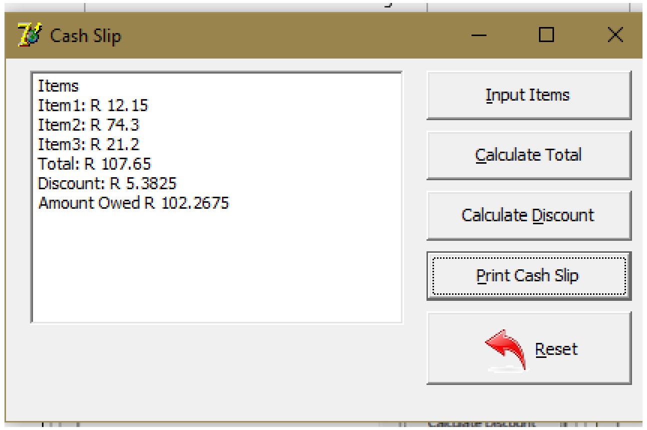
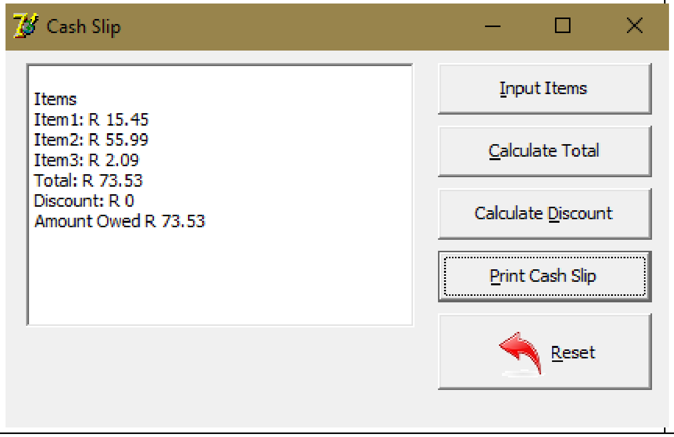
<table>
<tr><td><b>Question 1</b></td><td><b>Scenario:</b>
A client buys three items. If the total cost is more than R100 a discount of 5% is given. Write a program to print a cash slip.</td><td><b>[25]</b></td><td><b>Cognitive Demand/Comments</b></td></tr>
<tr><td colspan="3">Open the Delphi program <b><i>pCashSlip</i></b> in the Question1 folder. The form contains the following GUI components:
 </td><td><b>Learner data:</b>
Gr10_T2.zip</td></tr>
</table>

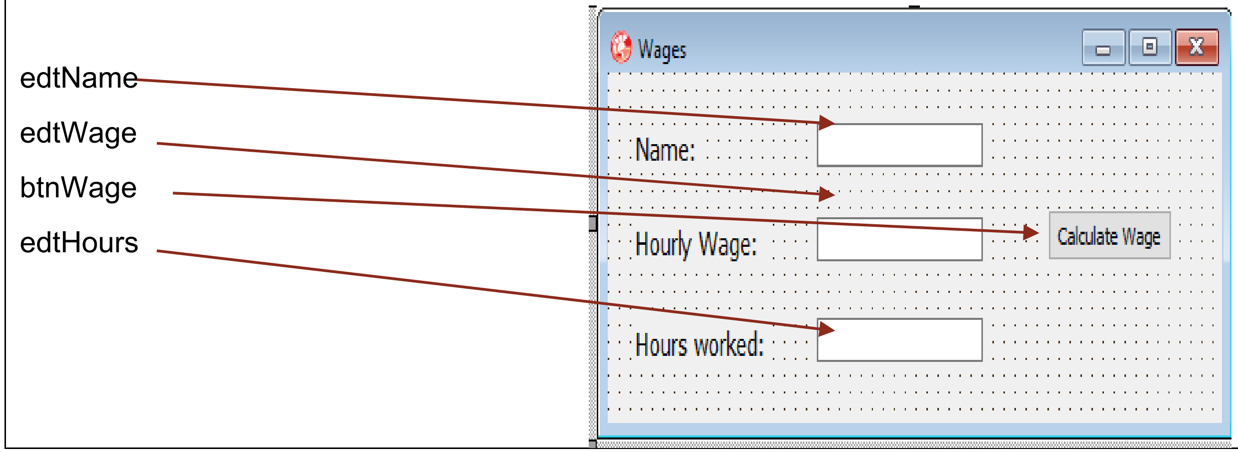
The following global variables were already created:

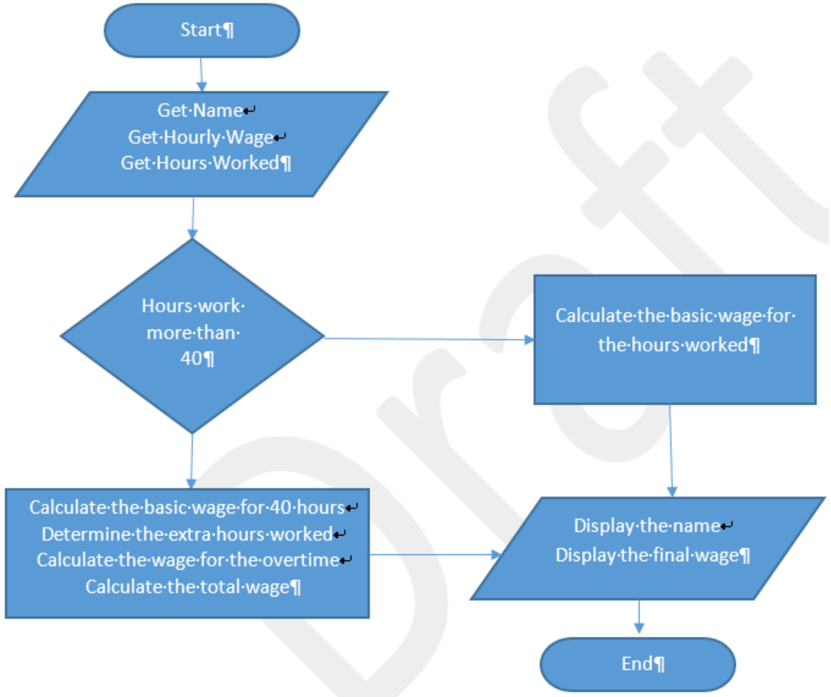
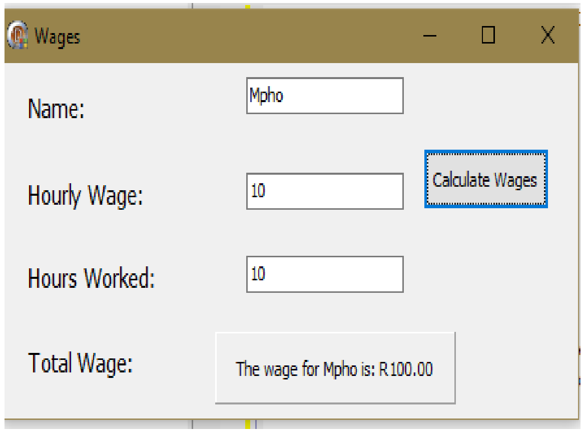
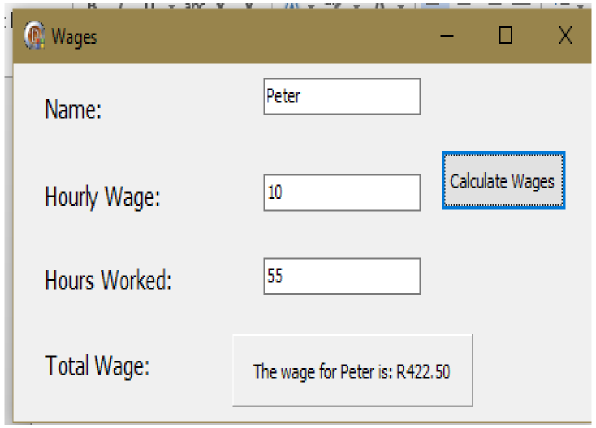
```
public
  sItem1, sItem2, sItem3  : String;
  rItem1, rItem2, rItem3  : Real;
  rTotal, rDiscount       : Real;
end;
```

1.1	Create an ***onClick*** event on ***bBtnReset.***		
1.1.1	Reset the following: • Clear the memo box • Set the three items to 0 (zero) • Set the discount and total to 0 (zero)	**(4)**	*Knowledge* Learners are said exactly what to do – direct explicit instructions Each can be done in isolation
1.2	Create an ***onClick*** event on ***btnInput***		
1.2.1	Enter the amounts of the three items by using input boxes	**(4)**	*Knowledge* Learners just need to use the inputbox. No application or understanding is needed
1.2.2	Display Items as a heading in the memo component  Items Item1: R 15.45 Item2: R 55.99 Item3: R 2.09 Cash Slip Input Items Calculate Total Calculate Discount Print Cash Slip Reset	**(1)**	*Knowledge* The learners are guided how the output must be.
1.2.3	Display the items and the amounts in the memo component as follows: **Notes:** • No tabs are used • You do not have to convert the numbers to currency.	**(3)**	

1.3	Create an ***onClick*** event on ***btnCalcTot.***		
1.3.1	Calculate the total by adding the three items.	**(2)**	*Knowledge* Learners are told how to calculate the total.
1.3.2	Display the total on the memo component.  Cash Slip Input Items Calculate Total Calculate Discount Print Cash Slip Reset Items Item1: R 15.45 Item2: R 55.99 Item3: R 2.09 Total: R 73.53	**(2)**	*Knowledge* The learners are guided how the output must be.
1.4	Create an ***onClick*** event on ***btnCalcDiscount.***		
1.4.1	Calculate the discount of 5% if the total is more than R100 and display the new discount.  Cash Slip Input Items Calculate Total Calculate Discount Print Cash Slip Reset Items Item1: R 12.15 Item2: R 74.3 Item3: R 21.2 Total: R 107.65 Discount: R 5.3825  Cash Slip Input Items Calculate Total Calculate Discount Print Cash Slip Reset Items Item1: R 15.45 Item2: R 55.99 Item3: R 2.09 Total: R 73.53 Discount: R 0	**(8)**	*Understanding and application* Learners need to understand how the discount is calculated and it must be subtracted from the total. Learners need to apply their knowledge of an IF-statement to this
1.5	Create an ***onClick*** event on ***btnPrintCashSlip.***		

1.5.1	Calculate the amount owed	**(1)**	*Knowledge*
	Test your output with discount:  Cash Slip Items Item1: R 12.15 Item2: R 74.3 Item3: R 21.2 Total: R 107.65 Discount: R 5.3825 Amount Owed R 102.2675 Input Items Calculate Total Calculate Discount Print Cash Slip Reset Test your output without discount:  Cash Slip Items Item1: R 15.45 Item2: R 55.99 Item3: R 2.09 Total: R 73.53 Discount: R 0 Amount Owed R 73.53 Input Items Calculate Total Calculate Discount Print Cash Slip Reset		

Question 2	**Scenario:** A worker receives a weekly wage. The wage is calculated by multiplying the number of hours with the hourly remuneration. Hours above 40 are overtime and the wage increases to one and a half of the normal wage.	**[20]**	**Cognitive Demand/Comments**
Open the Delphi program ***pWage*** in the Question2 folder. The form contains the following GUI components:  edtName edtWage btnWage edtHours Wages Name: Hourly Wage: Calculate Wage Hours worked:			Learner data: Gr10_T2.zip

2.1.	Use the following flow chart to do the coding on the Calculate Wage button.		
	 Start Get Name Get Hourly Wage Get Hours Worked Hours work more than 40 Calculate the basic wage for the hours worked Calculate the basic wage for 40 hours Determine the extra hours worked Calculate the wage for the overtime Calculate the total wage Display the name Display the final wage End		*Understanding*
	Test your coding with the following input  Name: Mpho Hourly Wage: 10 Hours Worked: 10 Total Wage: The wage for Mpho is: R100.00  Name: Peter Hourly Wage: 10 Hours Worked: 55 Total Wage: The wage for Peter is: R422.50		

Question 3		**[19]**	**Cognitive Demand/Comments**
Write a program to convert a binary number to a decimal number. The user must be able to enter any binary number. An alternative method to do this conversion was devised by William George Horner. Examine the following example explaining Horner's method. To convert the binary number to 11011001 to a decimal number the following is done: 0 x 2 + **1** = 1 1 x 2 + **1** = 3 3 x 2 + **0** = 6 6 x 2 + **1** = 13 13 x 2 + **1** = 27 27 x 2 + **0** = 54 54 x 2 + **0** = 108 108 x 2 + **1** = 217 Thus: $11011001_2 = 217_{10}$			*Problem solving – analysis/ evaluation/ creation* The learner needs to analyse the example given and identify the pattern to solve the problem.

Memorandum

Question 1

1.1.1	procedure TfrmCashSlip.bBtnResetClick(Sender: TObject);
	begin
	memCashSlip.Clear;
	rItem1 := 0;
	rItem2 := 0;
	rItem3 := 0;
	rTotal := 0;
	rDiscount := 0;
	end;
		onClick event created ✓
		memo box cleared ✓
		One variable set to 0 ✓
		All other 4 variables set to 0 ✓
			(4)	

1.2.1	sItem1 := inputbox('Item1','Please type in your item 1',");
	sItem2 := inputbox('Item1','Please type in your item 2',");
	sItem3 := inputbox('Item1','Please type in your item 3',");
	rItem1 := StrToFloat(sItem1);
	rItem2 := StrToFloat(sItem2);
	rItem3 := StrToFloat(sItem3);
		Inputbox created ✓ assigned to a variable ✓
		Applied to the other two variables ✓
		StrToFloat used ✓
			(4)	

1.2.2	memCashSlip.Lines.Add('Items');	Heading displayed ✓	**(1)**	

1.2.3	memCashSlip.Lines.Add('Item1: R ' + sItem1);
	memCashSlip.Lines.Add('Item2: R ' + sItem2);
	memCashSlip.Lines.Add('Item3: R ' + sItem3);
		One item displayed ✓ with value ✓
		Applied to the other two items ✓
		Total displayed before discount ✓
			(4)	

1.3.1	rTotal := rItem1 + rItem2 + rItem3;	Assigned to rTotal ✓ Adding the three items ✓	**(2)**	
1.3.2	memCashSlip.Lines.Add('Total: R ' + FloatToStr(rTotal));	Display total ✓ FloatToStr used ✓	**(2)**	

1.4.1	If rTotal > 100 then
	begin
	rDiscount := rTotal * 0.05;
	rTotal := rTotal - rDiscount;
	end;
	memCashSlip.Lines.Add('Discount: R ' + FloatToStr(rDiscount));
		If statement used ✓
		Correct ✓ Criteria used ✓ rTotal * 5% ✓
		Assigned to rDiscount ✓
		Discount subtracted from Total ✓
		Assigned to rTotal ✓
		Discount displayed ✓
			(8)	

1.5.1	memCashSlip.Lines.Add('Amount Owed R ' + FloatToStr(rTotal));	Total owed displayed ✓	**(1)**	

Question 2

	procedure TfrmWage.Button1Click(Sender: TObject);
	begin
	sName := edtName.Text;
	rHourlyWage := StrToFloat(edtWage.Text);
	iHoursWorked := StrToInt(edtHours.Text);
		Input assigned to one variable ✓
		Input correctly assigned (including casting) ✓
		to all three variables ✓
			(3)	

	if ✓ iHoursWorked > 40 ✓ then
	begin
	rBasicWage := rHourlyWage * 40;
	iOverTimeHours := (iHoursWorked - 40);
	rOverTimeWage := iOverTimeHours * 1.5;
	rTotalWage := rBasicWage + rOverTimeWage;
	end
	else
	rTotalWage := rHourlyWage * iHoursWorked;
		If statement used ✓
		Correct criteria used ✓
		Calculate basic wage: Hourly wage ✓ * 40 ✓
		Calculate the overtime: hours work ✓ - 40 ✓
		Calculate the overtime wage: overtime hours ✓ * 1.5 ✓
		Calculate total wage: Adding✓ basic wage to overtime wage ✓
		Else used ✓
		Calculate totalWage: Multiplying ✓ hourly wage with hours worked ✓
			(13)	

	pnlTotWage.Caption := 'The wage for ' + sName + ' is: ' + FloatToStrF (rTotalWage, ffCurrency,6,2);	Output on panel ✓ Name included ✓ FloatToStrF used ✓ ffCurrency used ✓	**(4)**	

Question 3

3.1.1	Algorithm design: IPO of flowchart	Input ✓✓ Processing ✓✓ Output ✓✓	**(6)**	

	Recommended solution
	procedure TForm1.btnConvertClick(Sender: TObject);
	var
	sBinary: string;
	i, digit, iDecimal: integer;
	begin
		Recommended marks
		One variable declared for string✓
		At least 1 number variable declared for integer✓
		Input for the binary number ✓
		Determine the length of the binary number
			(13)	

	sBinary := edtBinary.text;
	iDecimal := 0;
	for i := 1 to length(sBinary) do
	begin
	digit := strtoint(sBinary[i]);
	iDecimal := iDecimal * 2 + digit;
	end;
	edtDecimal.text := inttostr(iDecimal);
	end;
		✓
		Extract each char ✓ from the string ✓ and convert to integer ✓ by using a for loop ✓
		Assign character to digit ✓
		Formula assigned to decimal ✓
		Use the answer of first formula (digit) ✓
		Loop is repeated according to the length of the binary number ✓
		Display the answer ✓		

	Alternative solution
	procedure TForm1.btnConvertClick(Sender: TObject);
	var
	sBinNum : String;
	iNum1, iNum2, iNum3, iNum4, iNum5, iNum6, iNum7, iNum8 : integer;
	iAns1, iAns2, iAns3, iAns4, iAns5, iAns6, iAns7, iAns8 : integer;
	begin
	sBinNum := edtBinNum.Text;
	iNum1 := StrToInt (sBinNum[1]);
	iNum2 := StrToInt (sBinNum[2]);
	iNum3 := StrToInt (sBinNum[3]);
	iNum4 := StrToInt (sBinNum[4]);
	iNum5 := StrToInt (sBinNum[5]);
	iNum6 := StrToInt (sBinNum[6]);
	iNum7 := StrToInt (sBinNum[7]);
	iNum8 := StrToInt (sBinNum[8]);
	iAns1 := 0 * 2 + iNum1;
	iAns2 := iAns1 * 2 + iNum2;
	iAns3 := iAns2 * 2 + iNum3;
	iAns4 := iAns3 * 2 + iNum4;
	iAns5 := iAns4 * 2 + iNum5;
	iAns6 := iAns5 * 2 + iNum6;
	iAns7 := iAns6 * 2 + iNum7;
	iAns8 := iAns7 * 2 + iNum8;
		Alternative marks
		Note to marker: This is an alternative solution that does not represent effective programming techniques. Thus learners can only get a maximum of 9 marks for this type of solution.
		One variable declared for string✓
		At least 1 number variable declared for integer✓
		Input for the binary number ✓
		Extract at least one char ✓ and convert to Integer ✓
		Formula assigned to decimal (iAns) ✓
		Use the answer on first formula ✓
		Formula repeated for other digits ✓
		Display the answer ✓
			(9)	This solution is penalised, i.e. a maximum mark of 9 (instead of 13) is awarded for this solution

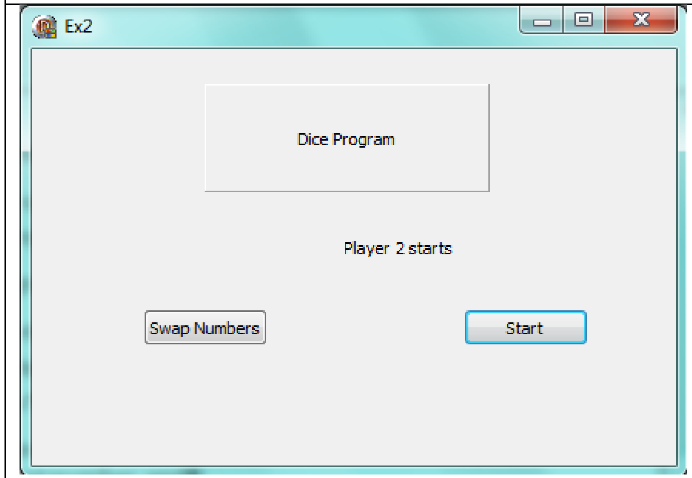
U

Grade 10 Term 3	**Marks:**	**[59]**	**Time:**	**60 minutes**

Instructions:

- You are required to answer ALL the questions.
- Use meaningful variable names that follow the correct naming conventions.
- Add comments if necessary to explain your code.
- Save all you work in the provided exam drive.
- Rename the gr. 10 data files folder by adding your name at the end.

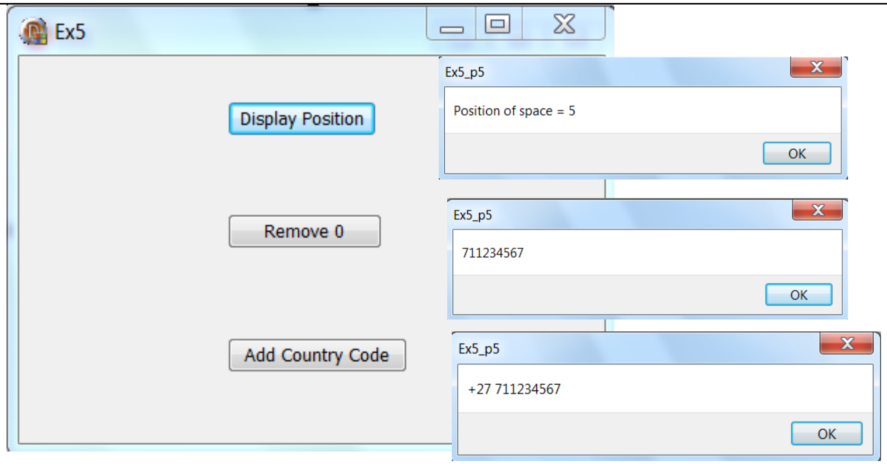
Question 1	Dice program	[8]	Cognitive Demand/Comments
	A simple dice program simulates the throw of a dice. When the program starts ([Start]-button is clicked), each of two players is assigned a number from 1 to 6. The number for Player 1 is assigned to the variable, Player1 and the number for Player 2 is assigned to the variable, Player2 (***You do not need to code this as this part is already coded and provided***). The player with the biggest number 'wins' and starts the game. ***Note: Assume that players will not be assigned the same number.***		**Learner data:** Ex2 (2).zip The code for each button (1.1 and 1.2) can be coded in isolation

	 Ex2
	Dice Program
	Player 2 starts
	Swap Numbers
	Start **Note:**
	Random numbers are assigned to the variables using a separate unit, Dex2, which is hidden.
			unit Dex2;
			interface
			var
			  Player1, Player2 : integer;
			implementation
			begin
			  Player1 := random(6) + 1;
			  Repeat
			    Player2 := random(6) + 1;
			  Until Player1 <> Player2;
			end.

1.1	Complete the code for the [Start]-button that will display which player (Player 1 or Player 2) will start the game as follows: • Write an if-statement that will determine which player has the highest number and display the player with the highest number that will start the game: *Player 1 starts* if player 1 has the highest number or *Player 2 starts* if player 2 has the highest number	**4**	*Knowledge* Recall of algorithm for determining the biggest/smallest one of two numbers
1.2	Sometimes players prefer to swap numbers before the 'winner' is determined. When the [Swap]-button is clicked, complete the code that will swap the numbers of the two players.	**4**	*Knowledge* Recall of the algorithm that swaps two values

Question 2	**Telephone number**	**[9]**	**Cognitive Demand/Comments**

	Open the project ***Ex5_P5***
	The variables, NameSurname and CellNumber are already declared.
	When the program is executed, values are assigned to the above variables, i.e. the following code is provided:
	procedure TfrmEx5.FormCreate(Sender: TObject);
	begin
	  NameSurname := 'John Public';
	  CellNumber := '0711234567';
	end;
			Learner data:

2.1	When the [*Display Position*]-button is clicked, display the position of the space in the NameSurname variable, e.g. for ***John Public***, display 5	**4**	*Knowledge* Direct, explicit instructions – can be coded in isolation – one concept at a time
2.2	When the [*Remove 0*]-button is clicked, display the cellphone number without the leading 0 (zero), i.e. delete the leading 0 from the cellphone number, e.g. **0711234567** becomes **711234567**	**3**	
2.3	When the [*Add Country Code*]-button is clicked, display the cellphone number with the country code, i.e. add the country code, **+27**, with a space in front of the cellphone number, e.g. **711234567** becomes **+27 711234567**.	**2**	
	 Ex5 Display Position Remove 0 Add Country Code Ex5_p5 OK Position of space = 5 711234567 +27 711234567		

Question 3 **Compound interest** **[15]**

Cognitive Demand/Comments

When you borrow money, you pay interest. If you borrow money for a period longer than one year, the interest of the previous year is added to the original loan amount and the interest for the subsequent year is calculated on the "new", bigger amount. So, the interest amount you pay grows annually as each year you pay "interest on interest".

This is called compound interest.

The table below illustrates how compound interest is calculated over a five-year period when borrowing R1 000.00 at 10% annual interest rate and how much you will owe after five years:

Year	Loan at Start	Interest Amount	Loan at End
0 (Now)	R1 000.00	(R1 000.00 × 10% =) **100.00**	R1 100.00
1	R1 100.00	(R1 100.00 × 10% =) **110.00**	R1 210.00
2	R1 210.00	(R1 210.00 × 10% =) **121.00**	R1 331.00
3	R1 331.00	(R1 331.00 × 10% =) **133.10**	R1 464.10
4	R1 464.10	(R1 464.10 × 10% =) **146.41**	R1 610.51
5	R1 610.51		

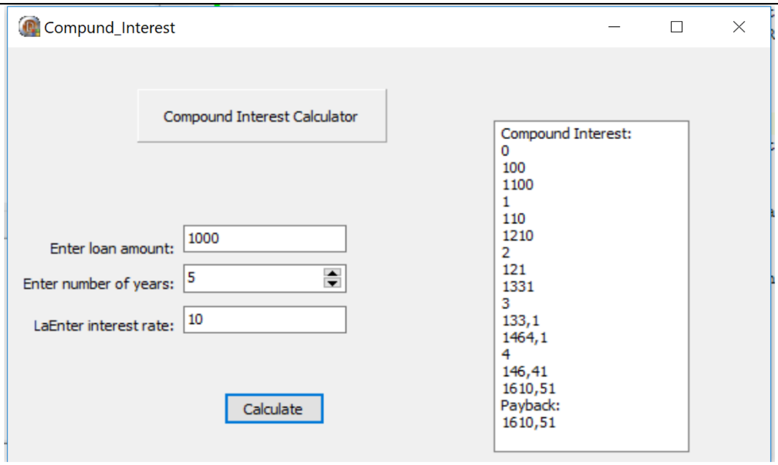
Use the Delphi program provided as well as the pseudo code below and convert it to a Delphi program that will calculate the annual loan at start, interest amount and loan at end as well as the final amount that needs to be paid back.

Learner data:

Gr10_T3_Q3.zip

Algorithm/Pseudo Code:

```
Initialise LoanAtStart, Year, Interest Rate and Interest
Repeat for Number of Years
        Interest ← e.g. 10% of Loan at Start
        LoanAtEnd ← Sum of Loan at Start and Interest
        Display Year, Loan at Start, Interest amount and Loan at End
        Assign new value to LoanAtStart
        Change year
Display Amount to be paid back
```



Enter loan amount: 1000

Enter number of years: 5

LaEnter interest rate: 10

```
Compound Interest:
0
100
1100
1
110
1210
2
121
1331
3
133,1
1464,1
4
146,41
1610,51
Payback:
1610,51
```

Understanding

Translating an algorithm from one form (of representation to another form)

Learner needs to interpret the algorithm/pseudo code and convert it to Delphi code

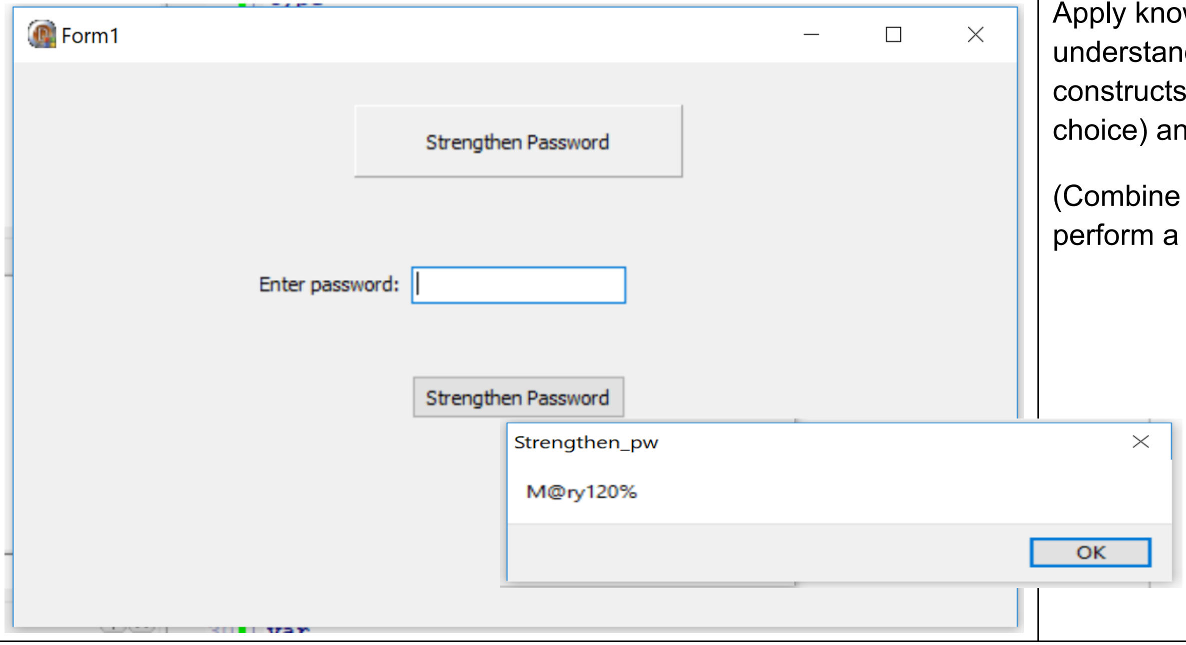
Input:

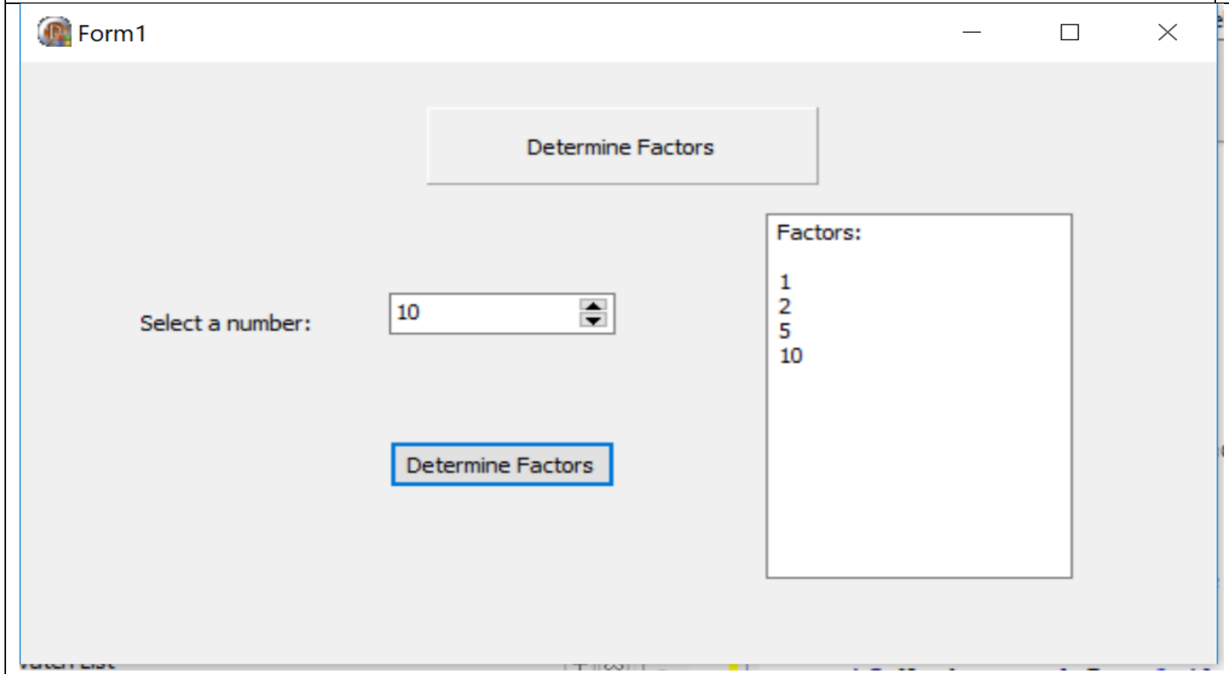
Loan amount, Number of years, Interest rate

Output

Year Number, Interest amount, Loan at End

Final amount to be paid back

Question 4	**Strengthening passwords**	**[9]**	**Cognitive Demand/Comments**
Many people use easy-to-guess passwords. Write a Delphi program that will strengthen a password by doing the following: • Replace all the occurrences of the lowercase letter a with the @ symbol • Replace all the occurrences of the number 0 with the capital letter O • Replace all the occurrences of the letter s or S and the number 5 with the % symbol The program must accept a password entered by the user and display the strengthened password using ShowMessage			**Learner data:** Gr10_T3_Q4.zip ***Application***
Input: A password, e.g. Mary1205 **Output** The strengthened password, e.g. M@ryO%	 Form1 Strengthen Password Enter password: Strengthen Password Strengthen_pw OK M@ry120%		Apply knowledge and understanding of code constructs (repetition and choice) and strings (Combine isolatable bits to perform a task)

Question 5	**Efficient programming**	**[6]**	**Cognitive Demand/Comments**
Open the program Factors. The program determines all the factors of an integer in ascending order. Rewrite the code to be more efficient, i.e. limit the number of times that the code within the list is executed. The program must still generate the same output.		**Learner data:** Factors.zip	*Analysis and Evaluation*
 Form1 Determine Factors Select a number: 10 Determine Factors Factors: 1 2 5 10			*Although the algorithm to determine the factors of a number is known (knowledge), changing it to be more effective requires analysis and evaluation*
Input: An integer number, e.g. 10	**Output** The factors of the integer number, e.g. 1, 2, 5 and 10		

Question 6	Write a Scratch program to display the ***next*** 10 numbers in a Tribonacci series with signature (0,1,1)	**[12]**	**Cognitive Demand/Comments**
The Tribonacci numbers start with three predetermined terms and each term afterwards is the sum of the preceding three terms. The Tribonacci numbers T_n are defined as follows: $T_1 = 0$, $T_2 = T_3 = 1$, and $T_n = T_{n-1} + T_{n-2} + T_{n-3}$ (n ≥ 3) In other words, each number is the sum of the previous three numbers. The first few Tribonacci numbers with signature (0,1,1) are: 0, 1, ,1, 2, 4, 7, 13, 24 … (**Note**: The first three terms are the signature. The signature may vary.)			*Analysis, Evaluation and Create* Free response question that includes analysis, pattern recognition, generalisation of pattern. Learner needs to relate, combine and integrate several concepts /code structures/ constructs to devise a ‘new’ algorithm or adapt existing ones to solve the problem. Example of a free response / open-ended question

Memorandum

Question 1

1.1	• if-statement used ✓
	• then part ✓
	• else part ✓
	• Correct output (if-statement correctly implemented) ✓
		procedure TfrmEx2.btnStartClick(Sender: TObject);
		begin
		  if Player1 > Player2 then lblWinner.Caption := 'Player 1 starts'
		        else lblWinner.Caption := 'Player 2 starts';
		end;
			4	

1.2	• Declare temporarily variable ✓
	• Assign first player to temporarily variable ✓
	• Assign second player to first player ✓
	• Assign temporarily variable to second player ✓
		procedure TfrmEx2.btnSwapClick(Sender: TObject);
		var
		  Temp : integer;
		begin
		  Temp := Player1;
		  Player1 := Player2;
		  Player2 := Temp;
		end;
			4	

Question 2

2.1	begin
	 ShowMessage ('Position of space = ' + IntToStr ✓ (Pos ✓ (' '✓,NameSurname✓)));
	end;
		4	

2.2	begin
	 Delete ✓ (Cellnumber, 1✓,1 ✓);
	 ShowMessage(Cellnumber);
	end;
		3	

2.3	begin
	 CellNumber := '+27 ' ✓+ Cellnumber ✓ // Insert ('+27 '✓,CellNumber,1 ✓); //
	 ShowMessage(CellNumber);
	end;
		2	

Question 3

3
Loop correctly done
	procedure TForm1.btnCalculateClick(Sender: TObject);
	var LoanStart, LoanEnd, InterestAmount, InterestRate : real;
	Years, YearNumber : integer;
	All variables declared ✓ Correctly declared
	begin
	LoanStart := strtofloat(edtLoanAmount.Text);
	Years := sdtYears.Value;
	InterestRate := strtofloat(edtInterestRate.Text);
	All input correctly initialised ✓and converted✓
	YearNumber := 0; ✓
	Repeat
	InterestAmount := InterestRate/100 ✓ * LoanStart; ✓
	LoanEnd := LoanStart + InterestAmount; ✓
	memOutput.Lines.Add(inttostr(YearNumber));
	memOutput.Lines.Add(floattostr(InterestAmount));
	memOutput.Lines.Add(floattostr(LoanEnd));
	Output: All three output items Correctly converted ✓ Correctly added to memo ✓
	LoanStart := LoanEnd; ✓
	YearNumber := YearNumber + 1; ✓ //inc(YearNumber)
	Until YearNumber = Years; ✓
	memOutput.Lines.Add('Payback:');
	memOutput.Lines.Add(floattostr(LoanEnd)); ✓
	end;
		15	No output formatting required, e.g. currency or fixed number of decimal places

Question 4

```
procedure TForm1.btnPasswordClick(Sender: TObject);
var
  Password : string; \\(not necessary)       All variables used
  I : Integer;                                correctly declared ✓
begin
  Password := edtPassword.Text;  \\(not necessary)
  for ✓I := 1 to length(Password) ✓do
    begin
      case ✓Password[I] ✓of
        'a'  : Password[I] ✓:= '@';
        'O' : Password[I] := '0';          All other cases
        '5','s', 'S' : Password[I] := '%';  correctly done ✓
      end;
     end;
  Showmessage(Password); ✓ outside loop ✓
end;
end.
```

9

Could use if-statement/s instead of case

Question 5

	begin
	Number := sedNumber.Value;
	memFactors.Lines.Add('1'); ✓ before loop ✓
	for I := 2 ✓to Number div 2 ✓do
	begin
	if Number mod I = 0 then memFactors.Lines.Add(inttostr(I));
	end;
	memFactors.Lines.Add(inttostr(Number)); ✓ after loop ✓
	end;
		6	

Question 6

	Outside loop: Correct statements✓ Set sum to 0 ✓ Set Term1, Term2 and Term3 to 0,1,1 (signature) respectively✓ Repeat ✓10 times ✓ Set sum to adding ✓ Term1, Term2 and Term3 ✓ Set Term1 to Term2 ✓ Set Term2 to Term3 ✓ Set Term3 to Sum ✓ Display sum ✓ (Correct statements in loop ✓)	**12**	

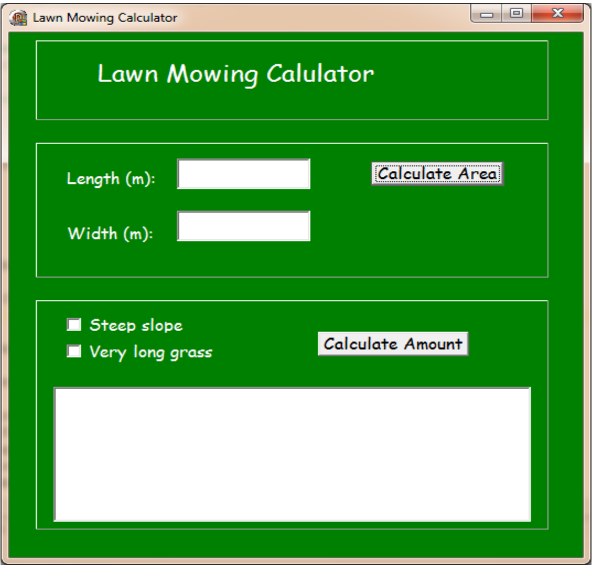
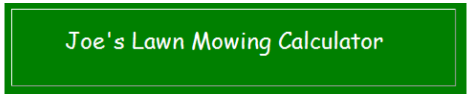
7.4 Grade 11 Term 1 - Practical Test

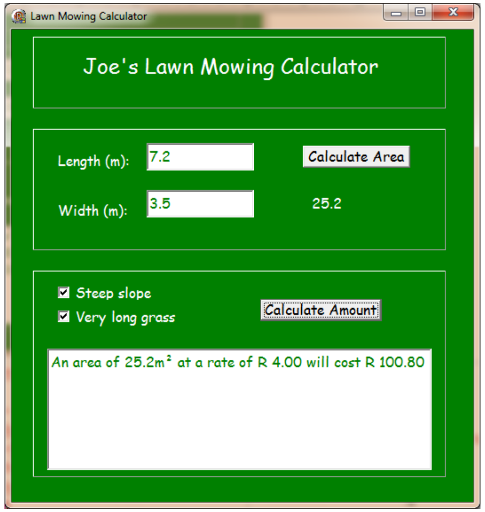
Task Description	**Comments**
This following task is a Grade 11 practical test that covers the practical content and skills for term 1: • GUI and components • Recall of basic code constructs and methods • Application of code constructs	Though assessment in Grade 11 should focus on Grade 11 content and skills, content and skills from Grade 10 (which is the foundation that is built upon in Grade 11) must also be revised continuously and may also form part of the test.

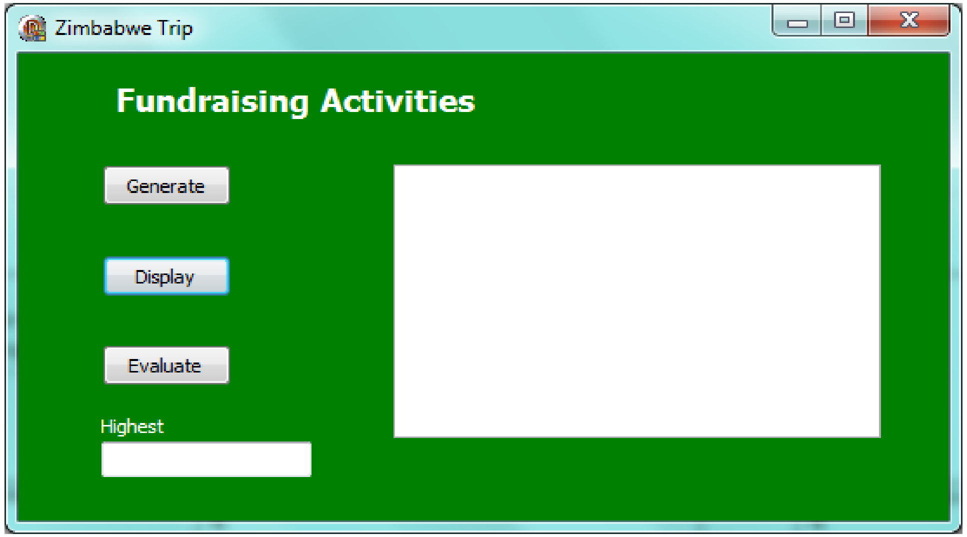
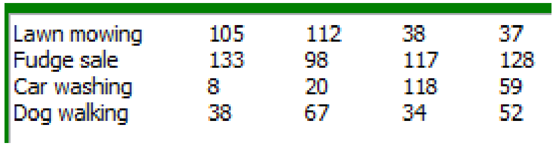
Cognitive Analysis

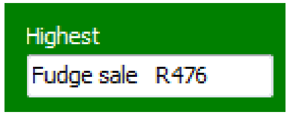
Cognitive Level		**Lower Order**	**Middle Order**	**Higher Order**	**Total**
Q1		**12**	**17**	**0**	**29**
	Q1.1	2			
	Q1.2.1	4			
	Q1.2.2	5			
	Q1.2.3		3		
	Q1.2.4	1			
	Q1.3.1		8		
	Q1.3.2		6		
Q2		**7**	**8**	**12**	**27**
	Q2.1	7			
	Q2.2		8		
	Q2.3			12	
Total		**19 (34%)**	**25 (45%)**	**12 (21%)**	**56**

<table>
<tr><td><b>Grade 11 Term 1</b></td><td><b>Marks:</b></td><td><b>[56]</b></td><td><b>Time:</b></td><td><b>75 minutes</b></td></tr>
<tr><td colspan="3"><b>Instructions:</b>
1. You are required to answer ALL the questions.
2. Use meaningful variable names that follow the correct naming conventions.
3. Add comments if necessary to explain your code.
4. Save all your work in the provided exam drive.
5. Rename the gr. 11 data files folder by adding your name at the end.</td><td colspan="2"><b>Scenario:</b>
You are planning to go on the provincial hockey tour to Harare, Zimbabwe, during the July holidays and your parents insisted that you earn and save some pocket money to use on the tour. You decided to create 2 apps to help you keep track of the money. The first app calculates some money you make with lawn mowing, and the second app is used to test which of your fundraising methods is the most effective.</td></tr>
</table>

Question 1	Lawn mowing calculator	[29]	Cognitive Demand/Comments
To earn some money you mow lawns for your neighbours and family. You charge a rate per m^2, but the rate changes depending on the slope of the lawn, the length of the grass, etc. Open the Delphi program ***frmMowLawn_p.dproj*** in the Question1 folder. The form contains the following GUI components: ○ pnlTop ○ lblHeading ○ pnlInput ○ edtLenght ○ edtWidth ○ btnCalcArea ○ lblArea ○ pnlOutput ○ chbSteepSlope ○ chbLongGrass ○ btnCalcAmount ○ memOut  Lawn Mowing Calculator Lawn Mowing Calulator Length (m): Width (m): Calculate Area Steep slope Very long grass Calculate Amount When the form is first opened pnlOutput is disabled (and therefore all the components on it are disabled too).			Example of structured question – consists of a stem (which describes a situation followed by a series of related questions). It leads learners through the question and they are less likely to stray from the subject or miss the point. Explicit, direct instructions – mostly no interpretation required – everything needed to answer the questions is immediately available to the learner. Learners' tasks are clearly indicated. Mostly, coding for each instruction can be done in isolation, however, care has to be taken that failure in one part does not affect the learners' answers in subsequent parts.
1.1	An ***onClick*** event has been added to the ***pnlTop*** – it therefore acts like big button and can be clicked anywhere. Add code to the given ***pnlTopClick*** method to add **your name** to the heading. When the panel is clicked anywhere inside the panel's frame, this method will be executed. Example of output of a learner named "Joe". To get the possessive apostrophe you have to add two single apostrophes to the code, e.g. Joe''s.  	2	*Knowledge* The first task is quite easy: • The onClick method is provided. • Concatenation of strings is quite easy. • An example is provided that shows what the finished task should accomplish.
1.2.1	Add an ***onClick*** event to ***btnCalcArea*** and create the necessary variables (a global variable ***rArea*** has been declared for your use already).	4	*Knowledge* Recall of a process practised

1.2.2	The button created in q1.2.1 must do the following: Get the length and width as input from the two text edits in ***pnlInput*** and assign the values to your variables.	**5**	*Knowledge* *Exactly the same context* as classroom-based exercise
1.2.3	Calculate the area of the lawn and display the area in ***lblArea***.	**3**	*Application* The learner has to apply his mathematical knowledge to the calculation of area
1.2.4	Enable ***pnlOutput***.	**1**	*Knowledge* Recall of how to enable
1.3.1	Calculate the amount of money earned. Your charges per m^2 are: • normal mowing R2.50 • on a steep slope R3.15 • very long grass R3.15 • very long grass on a steep slope R4.00	**8**	*Understanding & Application* Some thinking and application of concepts required. Learner needs to interpret GUI.
1.3.2	Display the output in ***memOut*** as in the example below. (Your values will differ depending on your input): **An area of 25.2m^2 at a rate of R4.00 will cost R100.80** **Note:** *Add “m^2” to the area – the ASCII code for ^{2} is 253.*  Lawn Mowing Calculator Joe's Lawn Mowing Calculator Length (m): 7.2 Width (m): 3.5 Calculate Area 25.2 ☑ Steep slope ☑ Very long grass Calculate Amount An area of 25.2m^2 at a rate of R 4.00 will cost R 100.80	**6**	*Application* Only the final output is given and all concatenation and formatting have to be figured out.

Question 2	**Fundraising Activities Tester**	**[27]**	**Cognitive Demand/Comments**
Over the past few weeks you have tried lawn mowing, fudge sales, car washing and dog walking as fundraising activities. You have tried each activity four times. You want to find out which of these fundraising activities was the most successful so that you can organise more of that rather than the other ones. To test your program you use random amounts at first. Open the Delphi program ***Fundraising_p.dproj*** in the Question2 folder. You will see the following GUI:  Zimbabwe Trip Fundraising Activities Generate Display Evaluate Highest			
2.1	**Button[Generate]:** Fill the four given arrays (arrLawnMowing, arrFudgeSale, arrCarWashing and arrDogWalking) with random amounts between 0 and 150 (whole numbers only).	**7**	*Knowledge* The global arrays are provided. Addressing the items in the arrays and randomising must be done just as in class.
2.2	**Button [Display]:** Display each activity together with the four amounts in neat columns, all on one line. **Example**: (NB. Your figures might be different as they are randomised numbers!) Lawn mowing 105 112 38 37 Fudge sale 133 98 117 128 Car washing 8 20 118 59 Dog walking 38 67 34 52	**8**	*Understanding & Application* The formatting has to be worked out.

2.3	**Button [Evaluate]:** Find the most successful fundraising activity which will be the one with the highest total revenue generated in the four trials. Display the highest total together with the activity. NB: You may declare any necessary variables/structures to complete this task. **Example**: (NB. Your outcome might be different as it depends on randomised numbers!)  Highest Fudge sale R476	12	*Evaluation* No guidance is given. The learner has to realise that the amounts first have to be added before the highest can be determined. No guidance is given on how to determine the highest total.

Memorandum

Question 1

1.1	lblHeader.Caption := 'Joe''s Lawn Mowing Calculator'; Concatenated the string✓ Set the string to the label✓ (Do **not** worry about the possessive apostrophe)		**2**	Do **not** worry about the possessive apostrophe.

1.2.1	procedure TfrmMowLawn.btnCalcAreaClick (Sender: TObject);
	var
	rLength, rWidth: real;
		Button Click event✓
		Vars declared: length✓ and width✓ real✓
			4	

1.2.2	begin
	rLength := StrToFloat(edtLength.Text);
	rWidth := StrToFloat(edtWidth.Text);
		Get length & width as input from the edits: strToFloat✓ (both)
		get length✓ and assign✓
		get width✓ and assign✓
			5	

1.2.3	rArea := rLength * rWidth;
	lblArea.Caption := FloatToStr(rArea);
		calculate area ✓
		convert to string✓
		display ✓
			3	

1.2.4	pnlOutput.Enabled := True;	enable pnlOutput ✓	**1**	

1.3.1	procedure TfrmMowLawn.btnCalcAmountClick (Sender: TObject);
	var
	rRate, rAmount: real;
	begin
	if (chbSlope.Checked AND chbLongGrass.Checked) then
	rRate := 4.0
	else
	if (chbSlope.Checked OR chbLongGrass.Checked) then
	rRate := 3.15
	else
	rRate := 2.5;
	rAmount := rRate * rArea;
		Button Click event✓
		Declare vars✓
		Check if BOTH checkboxes are checked✓
		Use R4.00 per ^{2}✓
		Check if only ONE of the checkboxes is checked✓
		Use R3.15 per ^{2}✓
		Otherwise use R2.50 per m^2✓
		Calculate amount earned (multiply)✓
			8	

1.3.2	with memOut.Lines do
	  begin
	    clear;
	    add('An area of ' + FloatToStrF(rArea, ffFixed, 6, 1) + 'm² at a rate of '+ FloatToStrF(rRate, ffCurrency, 6, 2) +' will cost '+FloatToStrF(rAmount, ffCurrency, 6, 2));
	  end;
		Sentence✓✓
		concatenation✓
		Unit for area✓
		FloatToStr✓
		Rate and amount earned formatted as currency✓
			6	

Question 2

2.1	procedure TForm1.btnGenerateClick(Sender: TObject);
	var
	  i: integer;
	begin
	  randomise;
	  // Lawn mowing - with loop
	  for i := 1 to 4 do
	    arrLawnMowing[i] := Random(150);
	  // Fudge sales
	  for i := 1 to 4 do
	    arrFudgeSale[i] := Random(150);
	  // Car washing
	  for i := 1 to 4 do
	    arrCarWashing[i] := Random(150);
	  // Dog walking - without loop
	  arrDogWalking[1] := Random(150);
	  arrDogWalking[2] := Random(150);
	  arrDogWalking[3] := Random(150);
	  arrDogWalking[4] := Random(150);
	end;
		Four values per activity✓ (with or without loop – see dog walking code)
		Random✓
		(150)✓
		Assign to global array✓
		
		Repeat for other three activities✓✓✓
			7	

2.2	procedure TForm1.btnDisplayClick(Sender: TObject);
	var
	  i: integer;
	  s: string;
	begin
	  randomise;
	  redDisplay.lines.clear;
	  // Lawn mowing
	  s := 'Lawn mowing';
	  for i := 1 to 4 do
	  begin
	    s := s + #9 + inttostr(arrLawnMowing[i]);
	  end;
	  redDisplay.lines.add(s);
	  // Fudge sales
	  s := 'Fudge sale';
	
	<u>Accept a solution without loop, e.g.</u>
	// Dog walking
	  s := 'Dog walking' + #9 + inttostr(arrDogWalking[1])
	  + #9 + inttostr(arrDogWalking[2])
	  + #9 + inttostr(arrDogWalking[3])
	  + #9 + inttostr(arrDogWalking[4]);
	  redDisplay.lines.add(s);
	end;
		Necessary variables✓
		
		Four times✓ (with or without loop – see dog walking code)
		Build string✓
		Using tabs (#9)✓
		Using the value from array✓
		Converting int to string✓
		Display✓
		
		Repeat for other three activities✓
			8	

2.3	procedure TForm1.btnEvaluateClick(Sender: TObject);
	var
	  i, iLmT, iFsT, iCwT, iDwT: integer;
	begin
	  iLmT := 0;
	  iFsT := 0;
	  iCwT := 0;
	  iDwT := 0;
	  for i := 1 to 4 do // instructions say must use a loop
	  begin
	    iLmT := iLmT + arrLawnMowing[i];
	    iFsT := iFsT + arrFudgeSale[i];
	    iCwT := iCwT + arrCarWashing[i];
	    iDwT := iDwT + arrDogWalking[i];
	  end;
	if (iLmT > iFsT) and (iLmT > iCwT) and (iLmT > iDwT)
	    then
	    edtHighest.Text := 'Lawn mowing ' + inttostr(iLmT);
	  if (iFsT > iLmT) and (iFsT > iCwT) and (iFsT > iDwT)
	    then
	    edtHighest.Text := 'Fudge sale ' + inttostr(iFsT);
	  if (iCwT > iFsT) and (iCwT > iLmT) and (iCwT > iDwT)
	    then
	    edtHighest.Text := 'Car washing ' + inttostr(iCwT);
	  if (iDwT > iFsT) and (iDwT > iCwT) and (iDwT > iLmT)
	    then
	    edtHighest.Text := 'Dog walking ' + inttostr(iDwT);
	end;
		Create variables for totals (individual variables or array (1..4)✓
		type integer✓
		Initialise totals✓
		before using them✓
		
		Add amounts from arrays (with or without loop) and store in variables/array for ONE✓
		Other three done✓
		All done correctly✓
		
		Compare to other three totals✓ using AND✓
		Display result✓ with activity✓
		Repeat for other three activities✓
		
		[If array used, the array can be sorted or standard algorithm for highest can be used; award
		loop(s)✓
		comparison✓
		swapping✓
		display result✓ with activity✓]
		
		DO NOT penalise any extra display of totals
			12	

7.5 Grade 11 Term 2 – June Examination – Practical Paper

Task Description	Comments
This following task is a Grade 11 practical June examination that focuses on the term 1 and term 2 content and skills: • Use and scope of variables • General programming constructs • String manipulation • Arrays • Text files	Although assessment in Grade 11 should focus on Grade 11 content and skills, content and skills from Grade 10 (which is the foundation that is built upon in Grade 11) must also be revised continuously and may also form part of the test.

Cognitive Analysis

Cognitive Level	**Lower Order**	**Middle Order**	**Higher Order**	**Total**
Q1	14	12	0	**26**
Q2	5	34	0	**39**
Q3	15		40	**55**
Total	**34 (28,3%)**	**46 (38,3%)**	**40 (33,3%)**	**120 (100%)**

Grade 11 Term 2 June Paper 2	Marks:	[120]	Time:	3 hours

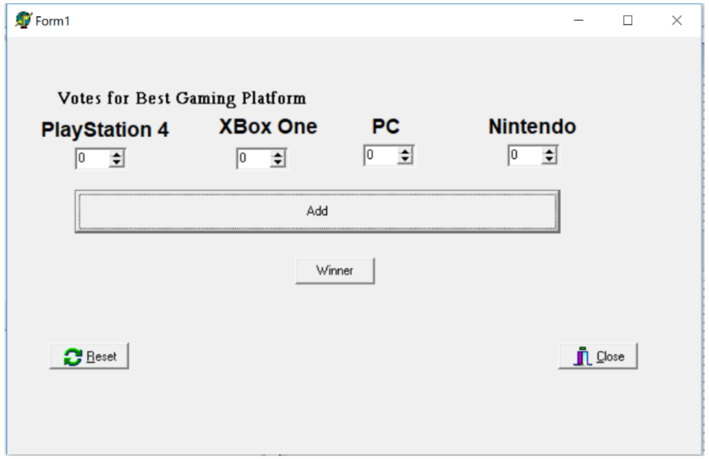
Instructions:

1. This question paper takes the form of an integrated scenario. If you are unable to complete any part of the paper and a following section requires you to make use of data derived from the previous section, continue with the coding regardless.
2. Save work regularly.
3. Manage your time carefully.
4. Do not do extra work outside of what is required.

Question 1	Voting	[26]	Cognitive Demand/Comments

The learners debate on the best platform for playing games. (PC, Play Station, Nintendo or Xbox). They decided to vote on this matter and requested you to create a Delphi program to administer the process.

An interface is provided in the ***Question 1 Data.zip*** folder. You are required to complete the coding.



The number of votes are entered using the spinner below the listed platforms (labels), e.g. PlayStation 4

- Add the following components, “PC” and “Nintendo”, each with ***SpinEdit*** to the form.
- Declare 2 arrays – one array to store the votes and one to store the platforms.
- When the form is activated, the arrays must be populated, i.e. the platforms assigned to the array that stores the platforms and the array that stores the votes must be initialised (values set to 0).

  Note: ***You must use arrays in the program and may not code in an alternative way.***
- Clicking the ***Add-button*** must add the number of votes entered for each platform to the vote counter for each platform and then reset the ***SpinEdit*** values to 0.
- The ***Reset-button*** must reset the number of votes obtained to allow the user to recount.
- When all the votes have been entered, the user clicks on the ***Winner-button*** and the name of the platform with the most votes is determined and displayed in the label **lblWinner**.

Learner’s response is restricted through the way the question is posed (must use arrays for platforms and votes)

Knowledge & Understanding

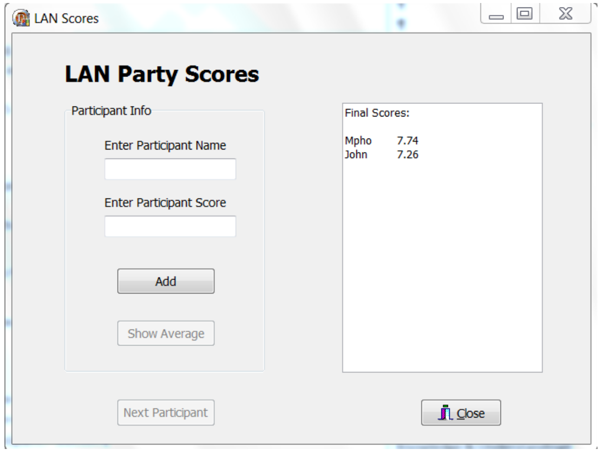
Question 2	LAN party software Your school is hosting a small LAN party in the computer lab. The organisers of the LAN party asked you to develop software to administer the process.	[39]	Cognitive Demand/Comments

An interface is provided in the ***Question 2 Data.zip*** folder. You are required to complete the coding.

Note: The "OnActivate event" of the form is **already** coded.

The organisers give you the following information:

- The number of participants is unknown
- There are 7 judges. Each judge awards a score out of 10. (e.g. 8.3)
- The highest and lowest scores for each participant are not considered when the final score is calculated.
- A participant's score is the average of the remaining 5 scores (without the highest and the lowest scores)



```
Mpho    7.74
John    7.26
```

Clicking the <Add>-button

- The user enters the participant's name and the 7 scores from the 7 judges.
- Reset the score button after each judge's score is entered to prepare for the next judge's score to be entered.
- After entering the 7th score, deactivate the <Add>-button and activate the <Show Average>-button.

When the <Show Average>-button is clicked, the following happens:

- Calculate the participant's final score as described above.
- Display the participant's name and average (rounded off to 2 decimals).
- Activate the <Next Participant>-button.

When the <Next Participant>-button is clicked, do the following:

- Prepare the edit boxes for the next participant's information to be entered.
- Reset the count and total global variables to ensure that the next participant's average will be calculated correctly.
- Deactivate the <Show Average>-button and the <Next Participant>-button
- Activate the <Add>-button.

Application
Task and process are clearly described – all the required information is immediately available to the learner from the way the problem is posed.

Familiar context (*situations with new circumstance or new elements*)

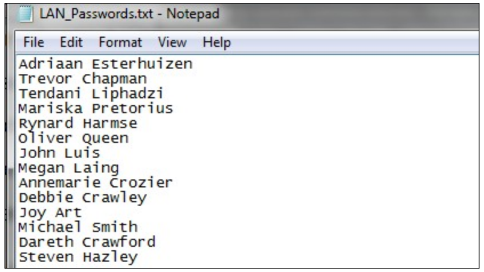
Knowledge & Understanding

Question 3	Creating passwords using data from a text file	[55]	Cognitive Demand/Comments

The organisers of the LAN party want to ensure fair competition and require you to create strong passwords for each participant.

You are provided a Delphi program and a text file, ***LAN_Passwords.txt***, in the ***Question_3 Data.zip*** folder.

The text file contains the first name and surname, separated by a space, of all participants, each on a separate line.



```
Adriaan Esterhuizen
Trevor Chapman
Tendani Liphadzi
Mariska Pretorius
Rynard Harmse
Oliver Queen
John Luis
Megan Laing
Annemarie Crozier
Debbie Crawley
Joy Art
Michael Smith
Dareth Crawford
Steven Hazley
```

<Get Data>-button:

Using first principles, load the data contained in the text file into the rich edit provided.

<Create Password>-button:

Use the data in the list box to create the passwords as follows:

Note: *If you were not able to load the data from the text file into the list box, click in the list box to load data (but remember you will not receive marks for anything that was not coded by yourself*).

- Shorten the name by deleting all the alphabet's "odd" numbered characters from the participant's name, e.g. Adriaan shorten to drn

Note: *An alphabet letter is defined as an "odd" letter if it is the 1st, 3rd, 5th, ... ,25th letter in the alphabet. For example, 'a', 'A', 'C', 'c', 'Y', 'y' are examples of "odd" letters, whilst 'B', 'b', 'X', 'x', 'Z', 'z' are examples of "even". letters*

Aa	Bb	Cc	Dd	Ee	Ff	Gg	Hh			Ss	Tt	Uu	Vv	Ww	Xx	Yy	Zz
1	2	3	4	5	6	7	8			19	20	21	22	23	24	25	26

- Build the password as follows:
  - Generate a random number, from 33 to 43 (both inclusive), then add the square of the random number to the shortened name, followed by
  - The ASCII code of the random number, from 33 to 43 (both inclusive), followed by
  - The length of the surname, followed by
  - The first letter of the surname

Example: The password for Tendani Liphadzi could be (depending on the randomised number): **Tndn38&8L.**
Remember the password will differ for each execution.

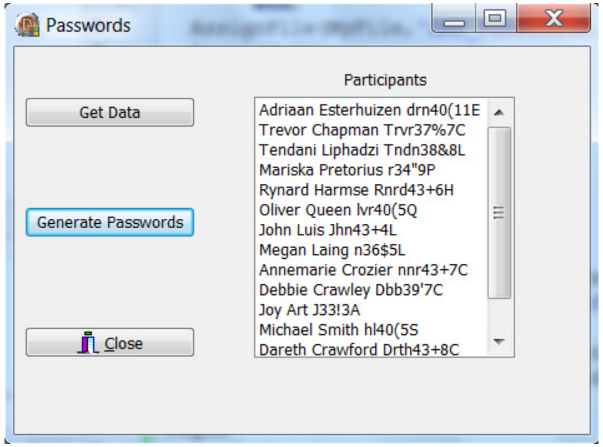
- Display the password next to the surname, separated by a space in the list box.

Knowledge

Reading text file (process knowledge) – exactly the same context as a classroom-based exercise – process/algorithm that learners were taught and have learned and practised before

Analysis

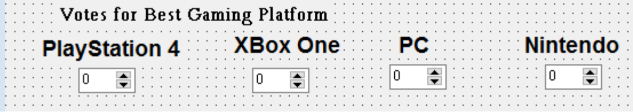
Requires reasoning and some analysis, developing a plan or sequence of steps (algorithm) to shorten the name – has some complexity

• Use an appropriate method and save the output displayed in the list box to a file named, PasswordTextFileExport.rtf.	
Example of output:  Passwords Get Data Generate Passwords Close Participants Adriaan Esterhuizen drn40(11E Trevor Chapman Trvr37%7C Tendani Liphadzi Tndn38&8L Mariska Pretorius r34"9P Rynard Harmse Rnrd43+6H Oliver Queen lvr40(5Q John Luis Jhn43+4L Megan Laing n36$5L Annemarie Crozier nnr43+7C Debbie Crawley Dbb39'7C Joy Art J33!3A Michael Smith hl40(5S Dareth Crawford Drth43+8C	

Memorandum

Question 1

	private
	arrCount : array[1..4] of integer;
	arrTypes : array[1..4] of string;
		Arrays declared ✓
		Correct data types used ✓
		Declared globally ✓
			3	Knowledge – 1
				Understanding – 2 (data types and globally declared)

	 Votes for Best Gaming Platform PlayStation 4 XBox One PC Nintendo 0 0 0 0	Placing of the 2 labels ✓ and the 2 spinedits ✓	**2**	Knowledge – 2

	procedure TForm1.FormCreate(Sender: TObject);
	var
	K : Integer;
	begin
	arrTypes[1] := 'Play Station4';
	arrTypes[2] := 'XBox One';
	arrTypes[3] := 'PC';
	arrTypes[4] := 'Nintendo';
	for K := 1 to 4 do
	arrCount[K] := 0;
	end;
		Correct event ✓
		Declaring the variable ✓
		Assigning platform ✓to all four ✓
		
		Initialise the array count:
		Loop✓ 4x✓
		Assign 0 ✓
			7	Knowledge – 7
				
				Initialising array elements

	procedure TForm1.btnOneClick(Sender: TObject);
	begin
	arrCount[1] := arrCount[1] + sedOne.Value;
	arrCount[2] := arrCount[2] + sedTwo.Value;
	arrCount[3] := arrCount[3] + sedThree.Value;
	arrCount[4] := arrCount[4] + sedFour.Value;
	sedOne.Value := 0;
	sedTwo.Value := 0;
	sedThree.Value := 0;
	sedFour.Value := 0;
	end;
		Increasing the number of votes for one array ✓
		Repeating the above for all four✓
		
		Clear SpinEdit ✓, done for all four ✓
			4	Understanding/Application – 2
				
				Knowledge – 2

	procedure TForm1.bmbResetClick(Sender: TObject); Var		**1**	Understanding – 1

	K : Integer;
	
	begin
	for K := 1 to 4 do
	arrCount[K] := 0;
	end;
		Repeat code from Form Create event ✓		

	procedure TForm1.btnWinnerClick(Sender: TObject);
	Var
	iWin, K : Integer;
	sWin : String;
	begin
	iWin := 0;
	
	for K := 1 to 4 do ✓
	if ✓arrCount[K] > iWin then ✓
	begin
	iWin := arrCount[K]; ✓
	sWin := arrTypes[K]; ✓
	end;
	lblWinner.Caption := 'The winner is: ' + sWin + ' with a total of: '+IntToStr(iWin);
	end;
		Declaring variables ✓, correct data type ✓
		Initialise variable ✓
		
		Using a loop to find the highest number of votes obtained ✓✓✓✓
		
		Display the answer in the label ✓
			9	*Knowledge* – 1 (declaring variables)
				Understanding – 2 (correct type and initialisation)
				
				Application – 6

			26	

Question 2

	procedure TfrmScores.btnAddClick(Sender: TObject);
	var
	rScore : real;
	begin
	rScore := StrToFLoat(edtScore.Text); ✓
	rTotal := rTotal + rScore; ✓
	inc(iCount); ✓ correct place ✓
	if ✓ iCount = 1 ✓then
	begin
	rHigh := rScore; ✓
	rLow := rScore; ✓
	end
	else ✓
	begin
	if ✓rScore > rHigh ✓ then
	rHigh := rScore; ✓
	if ✓ rScore < rLow ✓ then
	rLow := rScore; ✓
	end;
	edtScore.Clear; ✓
	edtScore.SetFocus; ✓
	if ✓ iCount = 7 ✓then
	begin
	btnAverage.Enabled := True; ✓
	btnAdd.Enabled := False; ✓
	end;
	end;
		Declare the variable with correct data type ✓
		Assign the values to the variable ✓
		Add scores ✓
		Increase the counter ✓✓
		
		Determine the highest and lowest score ✓✓✓✓✓✓✓✓✓✓✓
		
		Reset score edit after each score: clear ✓ and set focus✓ correct place✓
		
		Enable <average>-button ✓ and disable <Add>-button ✓after 7th score is entered✓
			22	Application – 21
				Knowledge – 1

	procedure TfrmScores.btnAverageClick(Sender: TObject);
	var
	  rAverage: Real; ✓
	begin
	  rAverage := (rTotal ✓- (rLow + rHigh) ✓/5; ✓
	  redScores.Lines.Add(edtName.Text ✓+ #9+ ✓ FloatToStrF ✓ (rAverage,ffFixed,5,2)); ✓
	  btnAverage.Enabled := False; ✓
	  btnNext.Enabled := True; ✓
	end;
		Declare variable with correct data type ✓
		Determine the average of the totals ✓✓✓
		Display the name of the participant and the average ✓✓✓✓
		Deactivate (Show Average) and activate (Add to Total) the two buttons ✓✓
			10	Application – 10

	procedure TfrmScores.btnNextClick(Sender: TObject);
	begin
	  edtName.Clear; ✓
	  edtName.SetFocus; ✓
	  edtScore.Clear; ✓
	  rTotal := 0; ✓
	  iCount := 0; ✓
	  btnNext.Enabled := False; ✓
	  btnAdd.Enabled := True; ✓
	end;
			7	Application – 3
				(preparing edit components for next participant – not explicitly instructed)
				Knowledge – 4
				Explicit instructions and can be done in isolation

			39	

Question 3

	procedure TfrmPasswords.btnDisplayClick(Sender: TObject);
	var
	MyFile: TextFile;
	sOneLine : string;
	iPos: Integer;
	begin
	if FileExists('LAN_Passwords.txt') = false then
	begin
	ShowMessage('File does not exist');
	exit;
	end;
	
	AssignFile(MyFile,'LAN_Passwords.txt');
	Reset(MyFile);
	
	while Not EoF(MyFile) do
	begin
	Readln(MyFile,sOneLine);
	redOutput.Lines.Add(sOneLine);
	end;
	CloseFile(MyFile);
	end;
		Declaring the Text file ✓
		Declaration for the variables and the correct data types used ✓✓
		
		Test if the text file is available ✓✓✓✓
		(Try – except can also be used)
		
		Opening the text file to read from it ✓✓✓✓✓✓✓
		
		Close the text file ✓
			15	Knowledge – 15

	<Create Passwords>-button
	Procedure TfrmLANParty.btnPasswordsClick(Sender: TObject);
	Const sAlpha = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'; //given
	var
	sLine, sName, sSurname, sPassWord : string;
	iCount, iPos, K, L, iLength, iCode : integer;
	bFind : boolean;
	begin
	Randomise; //given
	For ✓ iCount := 0 to lstData.Count – 1 ✓ do
	begin
	sLine := lstData.Items[iCount]; ✓
	iPos := pos ✓ (' ', sLine); ✓
	sName := copy ✓ (sLine,1,iPos-1); ✓
	delete ✓ (sLine,1,iPos); ✓
	sSurname := sLine; ✓
	iLength := length(sName); ✓
	sPassWord := ''; ✓
	For ✓K := 1 to iLength ✓do
	
	Initialise – both correct place
			40	Analysis – 40

	begin
	bFind := false; ✓
	L := 1; ✓
	While ✓ (not(bFind)) ✓and (L <= 26) ✓do
	begin
	if uppercase ✓ (sName[K]) ✓ = sAlpha[L] ✓
	then begin
	bFind := true; ✓ correct place ✓
	if ✓L mod 2 = 0 ✓then sPassWord:= sPassWord + sName[K] ✓;
	end;
	inc(L); ✓ correct place ✓
	end;
	end;
	iCode := Randomrange ✓ (33,44); ✓
	sPassWord := sPassWord ✓+ IntToStr (iCode) ✓+ char(iCode) ✓+ IntToStr (Length(sSurname)) ✓+ sSurname[1] ✓;
	sLine := lstData.Items[iCount] + ' ' + sPassWord; ✓
	lstData.Items[iCount]:= sLine; ✓ (Replaces “old” item)
	end;
	lstData.Items.SaveToFile ('PasswordTextFileExport.txt') ✓
	end;
		Needs to build the shortened name – cannot use delete		

			55	

7.6 Grade 11 Term 3 – Practical Test

Task Description	**Comments**
This following task is a Grade 11 practical test that covers the practical content and skills for term 3, which focuses on databases and arrays. The test covers • General programming concepts, including GUI components, and calculations, basic string manipulation • Arrays: ○ Data manipulation • Database: ○ Basic principles	Though assessment in Grade 11 should focus on Grade 11 content and skills, content and skills from Grade 10 (which is the foundation that is built upon in Grade 11) must also be revised continuously and may also form part of the test.

Cognitive Analysis

Cognitive Level	**Lower Order**	**Middle Order**	**Higher Order**	**Total**
Q1.1	2			**2**
Q1.2	5	8		**13**
Q1.3	2	12		**14**
Q1.4	9	5	15	**29**
Total	**18 (31%)**	**25 (43%)**	**15 (26%)**	**58**

Grade 11 Term 3	Marks:	[58]	Time:	1 Hour

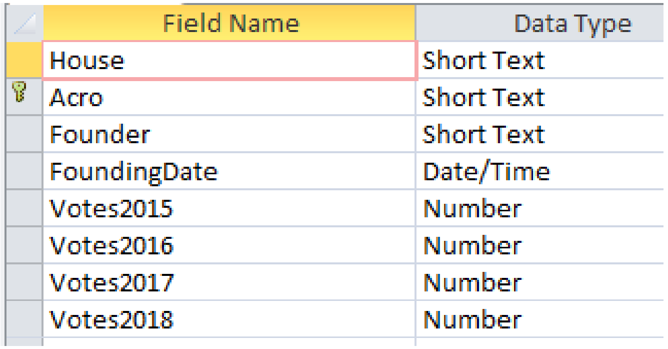
Instructions:

1. Answer ALL questions
2. Due to the nature of this test, you will not be allowed to leave the examination room before the end of the examination session.
3. Make sure that your name and surname appear as a comment in the first line of code.
4. At the end of the test you must hand in the disk/CD with all your work saved on it AND make a printout of your solution.
5. You have been provided with the following data files:

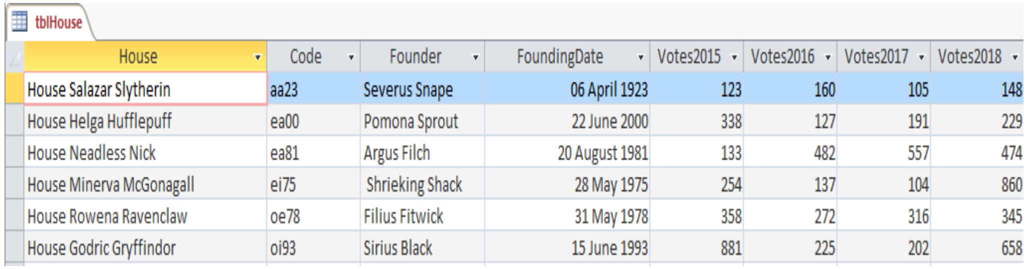
FOLDER: HouseElections

Question1Folder

q1_p.dproj



Field Name	Data Type
House	Short Text
Acro	Short Text
Founder	Short Text
FoundingDate	Date/Time
Votes2015	Number
Votes2016	Number
Votes2017	Number
Votes2018	Number



tblHouse

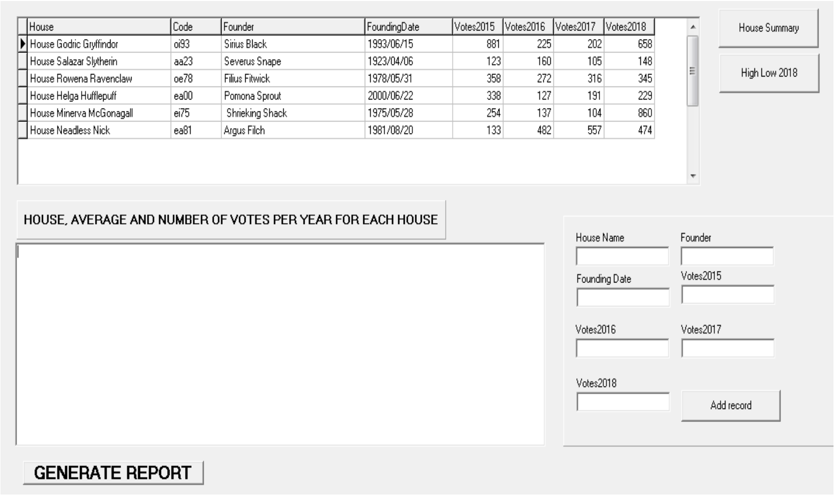
House	Code	Founder	FoundingDate	Votes2015	Votes2016	Votes2017	Votes2018
House Salazar Slytherin	aa23	Severus Snape	06 April 1923	123	160	105	148
House Helga Hufflepuff	ea00	Pomona Sprout	22 June 2000	338	127	191	229
House Neadless Nick	ea81	Argus Filch	20 August 1981	133	482	557	474
House Minerva McGonagall	ei75	Shrieking Shack	28 May 1975	254	137	104	860
House Rowena Ravenclaw	oe78	Filius Fitwick	31 May 1978	358	272	316	345
House Godric Gryffindor	oi93	Sirius Black	15 June 1993	881	225	202	658

Scenario

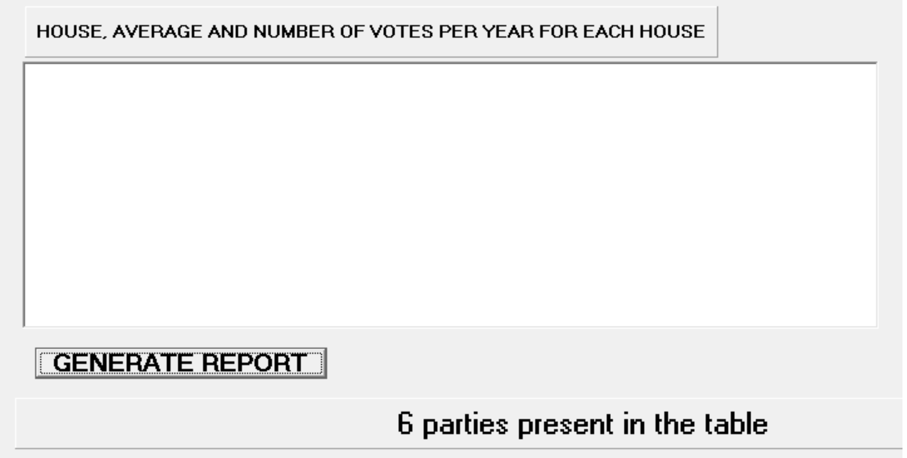
You are part of an Election Committee to extract data and assist with the elections for the house members of the house.

NOTE:

- A zero for any house's votes field is an indication that the house did not participate in that specific election.
- There is no need to use functions and procedures (procedural programming) in the completion of this question as the focus here is on Delphi-database interaction.
- Make a copy of the database prior to adding or removing any records as both these actions bring about permanent changes in the database.



House	Code	Founder	FoundingDate	Votes2015	Votes2016	Votes2017	Votes2018
House Godric Gryffindor	oi93	Sirius Black	1993/06/15	881	225	202	658
House Salazar Slytherin	aa23	Severus Snape	1923/04/06	123	160	105	148
House Rowena Ravenclaw	oe78	Filius Fitwick	1978/05/31	358	272	316	345
House Helga Hufflepuff	ea00	Pomona Sprout	2000/06/22	338	127	191	229
House Minerva McGonagall	ei75	Shrieking Shack	1975/05/28	254	137	104	860
House Neadless Nick	ea81	Argus Filch	1981/08/20	133	482	557	474

Question 1.1	Open q1_p.dproj	[2]	Cognitive Demand/Comments
Generate Report-button (btnReport) Write code to determine the number of political parties of which the details are present in tblHouse. Display this number on the panel named pnlResult as part of a user-friendly message.  6 parties present in the table			Knowledge

Question 1.2		[13]	Cognitive Demand/Comments

House Summary Button (btnHouseSummary)

- Write code to read all the fields from the database table (tblHouse).
- Your code has to calculate the average number of votes each house has received across all four completed elections.
- Display the house name, the average number of votes as well as the number of votes for each of the four elections neatly in columns under appropriate headings in the output component called redOutput. See screenshot of interface for details

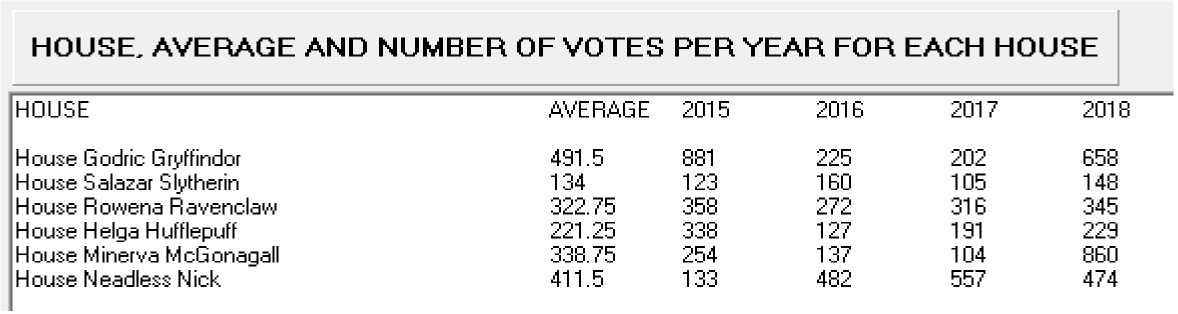
Knowledge (5)

Understanding (5)

The other five marks are interpretation of the question that leads to understanding the programming structures needed.

Application (3)

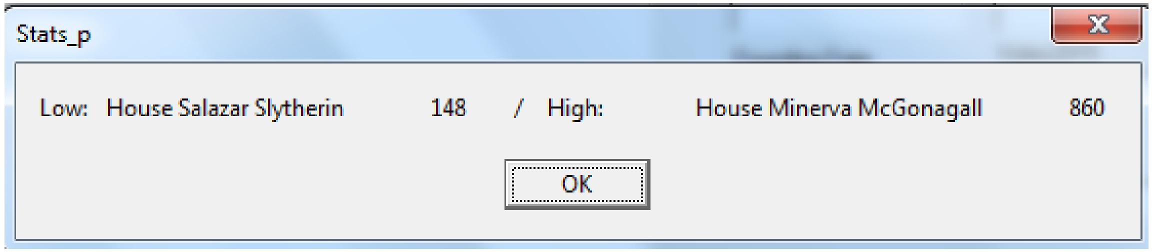
Needs prior knowledge to apply the programming statement.

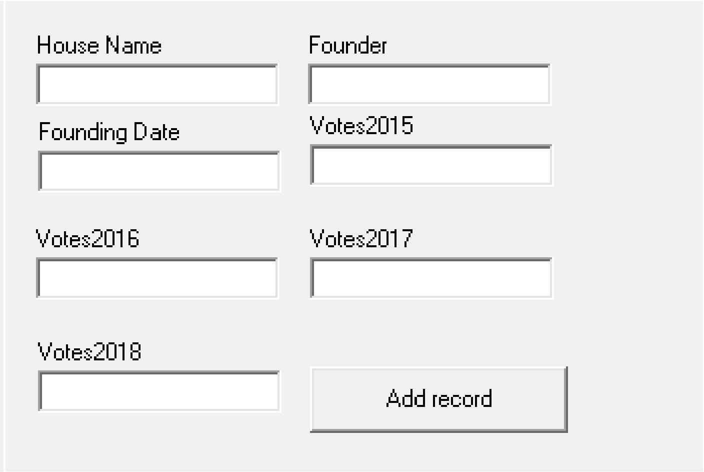


HOUSE, AVERAGE AND NUMBER OF VOTES PER YEAR FOR EACH HOUSE

```
HOUSE                                     AVERAGE   2015      2016      2017      2018

House Godric Gryffindor                   491.5     881       225       202       658
House Salazar Slytherin                   134       123       160       105       148
House Rowena Ravenclaw                    322.75    358       272       316       345
House Helga Hufflepuff                    221.25    338       127       191       229
House Minerva McGonagall                  338.75    254       137       104       860
House Neadless Nick                       411.5     133       482       557       474
```

Question 1.3		**[14]**	**Cognitive Demand/Comments**
High Low 2018-button • Declare two one-dimensional arrays called arrVotes and arrAccros respectively. You may assume that no more than 100 values will be stored in either of the arrays. • Read the number of votes stored in the 2018 field in the table into arrVotes2018. Read the house acronyms stored in the Acro field in the table into arrAcros. • Use the values stored in the two arrays to determine the house that obtained the highest number of votes in the 2018 elections as well the house that obtained the least number of votes in the same election. Use showmessage to display the following: • Low: House Name Votes attained by lowest house / • High: House Name Votes attained by highest house See screenshot.  Low: House Salazar Slytherin 148 / High: House Minerva McGonagall 860 			*Knowledge* (2) *Application* (5) Using ***known routines/ steps/ processes*** in order to complete a task. Application (7) The candidate needs to interpret and evaluate the question and screenshot that were provided to search for the highest and the lowest votes. It is also dependent on the previous programming section.

Question 1.4		**[29]**	**Cognitive Demand/Comments**
Add Record-button • Obtain the following data from the user to add to tblHouse. **House name: House Peter Pan** **Founder: Tinkerbell Hook** **Founding date: 22 November 2015** **Votes2015: 0** **Votes2016 120** **Votes2017: 240** **Votes2018 960**  House Name Founder Founding Date Votes2015 Votes2016 Votes2017 Votes2018 Add record • Use the house name and founding date in order to derive the code to be stored in the code field in the table. The code consists of the following: ○ Extract the middle name and find the first two vowels in the middle name. ○ Add the last two digits of the founding date to the extracted vowels, e.g. House Neadless Nick; 20 August 1981 – **ea81**			*Knowledge* (9) *Application* (5) Requires knowledge and understanding of steps/process/ isolatable bits. Analysis (15) Requires reasoning/ investigation/ developing a plan or algorithm; has some complexity. Completing task could have more than one possible approach. Organising component parts to achieve an overall objective.

Memorandum

Question 1

1.1	pnlResult.caption := inttostr(dmParty.tblHouse.RecordCount) ✓✓ + ' parties present in the table';		**2**	*Knowledge*: Basic string manipulation

1.2	procedure TForm1.btnHouseSummaryClick(Sender: TObject);
	var itot, inum: integer;
	   ravg : real;
	
	begin
	  redoutput.paragraph.tabcount := 5; //1 ✓
	  redoutput.paragraph.tab[0] := 200;
	  redoutput.paragraph.tab[1] := 250;
	  redoutput.paragraph.tab[2] := 300; //1 for all ✓
	  redoutput.paragraph.tab[3] := 350;
	  redoutput.paragraph.tab[4] := 400;
	
	  redoutput.lines.add('HOUSE' + #9 + 'AVERAGE' + #9 + '2015' + #9 + '2016' + #9 + '2017' + #9 + '2018'); //1 ✓
	  redoutput.lines.add('');
	
	  with dmparty do //1 ✓ not for with but for use of dmparty
	  begin
	    tblHouse.First; //1 ✓
	
	    while tblHouse.Eof = false do //1 ✓
	     begin
	     inum :=0; //1 ✓
	
	     if tblHouse['votes2015'] > 0 then inc(inum);
	     if tblHouse['votes2016'] > 0 then inc(inum);
	     if tblHouse['votes2017'] > 0 then inc(inum); //1 ✓ for all
	     if tblHouse['votes2018'] > 0 then inc(inum);
	
	      itot := tblHouse['votes2015'] + tblHouse['votes2016']+ tblHouse['votes2017']+ tblHouse['votes2018'] ; //1 ✓
	      ravg := roundto(itot/inum, -2); //2 ✓✓
		Basic programming structures that have been practised as is knowledge
		
		Needs prior knowledge to apply the programming statement – application
			13	*Knowledge* (5)
				
				Understanding (5)
				The other five marks are an interpretation of the question that leads to understanding the programming structures needed.
				
				Application (3)
				Needs prior knowledge to apply the programming statement.

	redOutput.lines.add(tblHouse['house']+ #9+ floattostr(ravg)+#9+ inttostr(tblHouse['votes2015'])+#9+ inttostr(tblHouse['votes2016'])+#9+inttostr(tblHouse['votes2017'])+#9+ inttostr(tblHouse['votes2018'])); //1 ✓
	
	tblHouse.Next; //1 ✓
	end; //while
	end;//do of with
	end;			

1.3	procedure TForm1.btnHighLow2018Click(Sender: TObject);
	var
	arrVotes : array[1..100] of integer; //
	arrAcros : array[1..100] of string; // 2 declaration and datatype✓✓
	k, l : integer;
	iSwap : integer;
	iCount : integer;
	sswap : string;
	
	begin
	iCount := 0; //1 ✓
	
	with dmParty do //given
	begin
	tblHouse.first; //given
	
	while not tblHouse.eof do //given
	begin
	inc(iCount); //1 ✓
	
	arracros[icount] := tblHouse['House']; //1 ✓
	arrVotes[icount] := tblHouse['Votes2018'];//1 ✓
	// 1 ✓ if instructions are placed in correct order inside the loop
	
	tblHouse.next; //given
	end; //while
	end; //with
		Basic array declaration – knowledge
		
		Understanding: Three instructions provided in correct order
			14	Knowledge (2)
				
				Understanding (5)
				Using ***known routines/ steps/ processes*** in order to complete a task.
				
				Application (7)
				The candidate needs to understand and apply the question and screenshot that were provided to search for the highest and the lowest votes. It is also dependent on the previous programming section.

	for k := 1 to icount - 1 do //1 ✓
	for l := k + 1 to icount do //1 ✓
	begin
	if arrVotes[k] > arrVotes[l] then //1 ✓
	begin
	iswap := arrvotes[k];
	arrvotes[k] := arrvotes[l]; //1 ✓ for swap
	arrvotes[l] := iswap;
	
	sswap := arrAcros[k];
	arracros[k] := arracros[l]; //1 ✓ for swap
	arracros[l] := sswap
	end; //if
	end; //for l
	
	showmessage('Low: ' + arracros[1] + #9 + ' ' + inttostr(arrvotes[1]) + #9 + ' / High: ' + #9 + arracros[icount] + #9 + inttostr(arrvotes[icount])); //1 ✓ for low and
	//1 ✓ for high
	end;
		Application
		
		Application		

1.4	procedure TForm1.btnAddClick(Sender: TObject); //[29] (1.4)
	var
	sCode, sname, sDate : string;
	iNum, K, iPos : Integer;
	begin
	
	with dmparty do //1
	begin
	tblHouse.Insert; //1
	
	tblHouse['house'] := edtName.Text; //1
	tblHouse['foundingdate'] := edtDate.Text;
	tblHouse['founder'] := edtFounder.Text;
	tblHouse['votes2015'] := edt2015.Text;
	tblHouse['votes2016'] := edt2016.Text; //2 for all
	tblHouse['votes2017'] := edt2017.Text;
	tblHouse['votes2018'] := edt2018.Text;
		Knowledge	[30]	Knowledge (9)
				
				Application (5)
				Requires knowledge and understanding of steps/process/ isolatable bits.
				
				Analysis (15)
				Requires reasoning/ investigation/ developing a plan or algorithm; has some complexity.
				Completing task could have more than one possible approach.
				Organising component parts to achieve an overall objective.

```
    sname := edtName.Text; //1
    iNum := 0;  //1
    K := 1;    //1
    sCode := ''; //1

    iPos := Pos( ' ',sName); //1
    Delete(sName,1,iPos);    //1
    iPos := Pos( ' ',sName); //1
    sname := Copy(sName,1,iPos-1);  //2

    While (K <= Length(sName)) AND (iNum < 2) do  //3
    begin
     if Upcase(sName[K]) IN ['A','E','I','O','U'] then  //3
      begin
        inc(iNum);   //1
        sCode := sCode + sName[K]; //2
      end;
     inc(K); //1
    end;
    sDate  :=  inputbox('enter  data  for','the  foundering  date',
'here');//1
      sCode := sCode + Copy(sDate,(length(sDate))-1,2); //3

    tblHouse['Code'] := sCode;   //1

  end;//with

  showmessage('New record has been saved.');
end;
```

Application

Analysis

Annexure A Question Guidelines

Written Papers

Clue Word	Meaning	Example	Notes
Analyse	Find the main ideas, how they are related and why they are important. Look for patterns or organisational principles. Identify problems or causes.	**Analyse** the correct use of word processing features in the following screenshot:	Break into parts/sections. Study each part – look at the detail. Find patterns/clues/problems/look for solutions. Examine and interpret the interrelationships and hierarchy of each as they relate to the whole. Infer from data.
Categorise	Group concepts/ideas that are similar/have the same characteristics/ functions or belong together. How should things be organised?	**Categorise** the following computer devices: keyboard, CPU, printer, mouse, hard drive, SSD according to their main function.	Which categories can describe the function/features? Understand relationships. Similar to **classify.**
Classify	Sort concepts/ideas according to categories or specific features showing which category or group they belong to.	**Classify** the following list of computer devices as input, output or storage devices.	Similar to **categorise.**
Comment on[1]	**Discuss**, **criticise**, or **explain** the meaning as completely as possible. Judge the value or appropriateness of something by applying proper criteria in a logical manner.	**Comment on** the use of a wireless network in the following case:	Fill in the gaps in terms of: What? Where? When? Who? How? Why? Highlight advantages and disadvantages/motivating or explaining why it is a good idea/not a good idea/what is wrong/which parts are good/how to improve, etc.
Compare	Show both the differences and the similarities of specific concepts.	**Compare** ROM with RAM	Find a relationship between two concepts. Which features/functions/uses are alike/differ?
Contrast/ distinguish	**Compare** by showing the differences or *unique* and **distinguishing** characteristics between concepts.	Contrast ROM and RAM or distinguish between a PAN and a HAN.	Identify the features of each, look for the ones that make them different. How does one thing differ from another? Which features/functions/uses are different?
Criticise	Give your **judgement** or reasoned opinion of something, showing its good and bad points.	**Criticise** the use of the keyboard when playing games.	What are the advantages and disadvantages? Why is it better to do it in a specific way rather than another way?

[1] Note that some clue words include or refer to other clue words

Clue Word	Meaning	Example	Notes
Define	Give a clear, concise **formal** meaning of a term or concept. The definitions should **distinguish** the concept from *related* terms/concepts. This is often a matter of giving a memorised definition.	**Define** phishing.	Give characteristics/features/functions. What is it?/What does it do/is it used for?/How does it look? Use short, concise descriptions of **main** features, focus on facts – sentence or two. The reader must know exactly what you are talking about and should not confuse it with something similar.
Describe	Write a detailed account or verbal picture in a logical sequence or story form. Give the main features by expanding the statement.	**Describe** phishing.	Tell a story about the issue in question. The reader should get a clear understanding of what it is/how it works or happens/why it happens/who or what is involved/where it is or takes place/when it happens. Show your understanding of the concept.
Diagram	Make a graph, chart or drawing. Be sure to label it and add a brief **explanation** if necessary.	**Diagram** a generic ICT system.	Visual representation of the main ideas/concepts/parts of something.
Differentiate	**Compare** two concepts and give the differences or unique and **distinguishing** characteristics between them	**Differentiate** between ROM and RAM.	Similar to **contrast/distinguish.**
Discuss	Write about a concept giving all the information. Present arguments for and against a point of view and reach a conclusion. The arguments must be supported with appropriate evidence or examples.	**Discuss** the use of a PAN in a home office.	**Analyse** the situation, look at the advantages and disadvantages, decide what will work/not work, explain why it will be a good idea or not a good idea. **Judge** the value.
Elaborate	Give more detail or **explain** or **justify** an answer or statement.	**Elaborate** on the advice given to Mr X regarding buying a laser printer.	
Evaluate	Give an opinion, supported by some expert opinions, of the truth or importance of a concept. Show the advantages and disadvantages.	**Evaluate** the merit of buying computer X for person Y.	Why would computer X be better than computer A? **Judge** the value of a situation/advice/concept.

Clue Word	Meaning	Example	Notes
Examine	Investigate, critically, appraise (**evaluate**, **judge)**, weigh up a subject in detail.	**Examine** the two methods.	Similar to **analyse.**
Explain	Give full reasons or **justifications** for something, or how and why something happened or works. Focus on interpretation of cause and effect. **Clarify**, **interpret**, and **elaborate**. Give reasons for differences of opinion or results, and try to **analyse** causes.	**Explain** how phishing works.	If you have trouble working out how to start answering a question that asks you to "explain", imagine you are telling a friend about the topic. You need to explain things about the topic such as: What is it, Where/When is it used/found, Why it is used/needed, How is it used/does it happen, What is the problem/consequences, Who is involved?
Identify	Recognise and **name** or provide a fact. Single out from other information.	**Identify** the port used to connect the printer.	Similar to **name.**
Illustrate	**Explain** or make clear by concrete examples, comparisons or analogies or sometimes with a visual element, such as a picture, drawing, figure, graph, or **diagram**.	**Illustrate** the information processing cycle.	Draw a **diagram** or give a real-life example.
Interpret	Give the **meaning** using examples or personal comments/understanding to make something clear.	**Interpret** the tone of the e-mail message given below: *I hate you when you do this* ☺	Deduce the meaning of something. Come to a conclusion about something. Translate knowledge into context to show understanding.
Justify	Give a statement of why you think something is the way it is. Give reasons for your statement or conclusion.	**Justify** the use of a table instead of tab stops in the following instance:	What is the situation? Why is the one better than the other?
List	Produce a list of words, sentences or comments. Give a short, brief statement, term or words.	**List** the types of information sources one could use when doing research.	Give *main* ideas, e.g. features, advantages, etc. Mostly recalling facts that you have learned and should know.
Motivate	Provide a reason or **justification** for an answer or statement.	**Motivate** the use of line and paragraph spacing when working in Word.	Why do you say so/suggest something, Why did it happen, Why is this better than that? Why should you do it this way?
Name/ mention	**Identify** the concept and provide a fact.	**Name** one storage device.	Similar to **list** or **identify.**
Outline	Give a general **summary**. It should contain a series of *main* ideas supported	**Outline** the information processing cycle.	Similar to **summarise**.

Clue Word	Meaning	Example	Notes
	by secondary facts. Show the organisation of the idea.		
Order	Provide a chronological or value-based answer by listing several items (terms or events in correct sequence).	**Order** the storage media according to their capacity.	List/arrange from small to large, few to many, slow to fast, write down the steps in order of execution, etc.
Prove	Show by using an argument or logic or fact that something is true.	**Prove** that the majority of people like X by looking at the following survey results:	Find *evidence* to support generalisation/statement/ conclusion, etc.
Relate	Show the *connection* between things, indicating how one causes or is like another.	**Relate** the following terms and explanations:	Match two concepts that belong together.
Review	Give a survey or summary in which you look at the important parts or major points and criticise if necessary. **Comment on** what is given.	**Review** Mr X's monthly computer maintenance tasks.	Briefly **analysing** each and **commenting** on them. Check if all necessary tasks are listed, frequency of tasks, etc. Make recommendations where necessary or suggest changes.
State	**Describe** the main points in specific terms. Use brief, clear sentences. Omit details or examples.	**State** the functions of the operating system.	
Suggest	**Analyse** or **examine** a problem/case and give possible reasons/ideas/solutions.	**Suggest** a computer configuration for Mr X.	What is the situation/what is given? What is needed? What would be the best? Why this? Consider needs, pros and cons.
Summarise	Give a brief, condensed description of the main ideas. Like developing an abstract.	**Summarise** the problems experienced in the following case:	Similar to **outline**.
Trace	Follow the development, progress or history of something, normally from the point of origin, typically in *chronological* order.	**Trace** the error in the spreadsheet calculations.	Why is the answer wrong? Find the problem.

Practical Papers

Clue Word	Meaning	Example	Notes
Call	Activate a function/method/routine in a program. Similar to **invoke.**	**Call** the function/method that will test if a word is a palindrome.	Write the function/method/routine and write code that will activate or implement it. Calling a routine consists of writing an instruction specifying the routine name and, optionally, parameters to activate it.
Change	Modify or adjust a structure or program/ program segment according to specific criteria or to produce a different outcome.	**Change** the loop structure so that it will stop when the user enters 'stop'.	
Code	Write program code to accomplish a task. Could be similar to **create** or **develop.**	**Code** a Scratch solution to calculate the cost for tiling a specific area.	
Complete	Use the code given and finalise a program/program segment to produce specific output or outcome or add code to finally accomplish a task.	Complete the Scratch program to provide the following output:..	Fill in the missing pieces (code). Determine where you have to add code.
Correct	Find the error, often through **tracing**, then change it to implement a program/program segment correctly.	**Correct** the loop structure so that it will provide the correct output.	**Trace** the problem or error and fix it. Alter the code to get the desired result.
Create	Write your own program from the problem statement/description given. Analyse, plan and produce a complete program/ program segment from a problem statement by combining elements ('building blocks') in the correct sequence and way to devise algorithm, solve a problem or satisfy a problem statement or produce the required outcomes.	**Create** a program that will solve the following problem: Convert a fraction to its simplest form, e.g. $\frac{8}{36}$ to $\frac{2}{9}$	Typically involves reasoning and all problem-solving steps. How would one solve this manually (pen-and-paper)? Which known algorithms could help solve the problem or parts of the problem? How must they be combined or adapted?
Debug	Find and remove errors in a program/ program segment. Similar to **correct**.	The following code is supposed to determine the average of 10 numbers but is not giving the correct output. **Debug** the program.	Repair or fix a structure, program/ program segment that is not functioning the way it should or that produces incorrect results. **Trace** the problem or error and fix it.

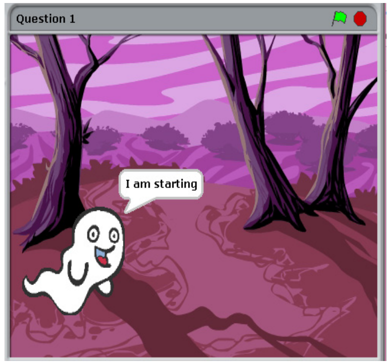
Clue Word	Meaning	Example	Notes
Develop	Plan, write and implement program code Similar to **create**.	**Develop** a Scratch program to convert Astronomic Units (AU) to miles and kilometres.	Analyse, determine requirements, plan, implement and test. Follow problem-solving steps (see page 19).
Execute	Run an existing program.	**Execute** the program and determine if the output is correct.	
Generate	Produce code or code segment(s) to solve a problem or perform a task. Similar to **develop** or **create.**	**Generate** Scratch code to calculate the VAT and the final price of a product.	
Implement	Put into effect or activate. Add to existing code to improve/add functionality.	**Implement** the following function/ method/code segment to extend the function of the program.	
Invoke	Call or activate a function/method or sub-routine.	**Invoke** a function/method that will validate the ID number entered.	
Re-factor	Rewrite existing code to make it better or more usable or improve the structure. Change existing code to accommodate added functionality.	**Re-factor** the method/function to provide for the following additional functionality…	It does not change the behaviour of code, i.e. the code must still perform the same action/provide the same output but in a better way or with other functionalities added.
Rewrite	Transform from one format/approach to another or to correct code or to implement a better/more effective solution or different method to accomplish a task.	**Rewrite** the program representation in the flow chart as a Scratch program.	
Trace	Follow the development, progress or history of something, normally from the point of origin, typically in *chronological* order or in the same sequence it is executed or implemented.	**Trace** the error in the program.	Why is the answer/output wrong? Find the problem.
Write	Code a computer program/program segment to perform a specific task or solve a problem.	**Write** a function/method/ set of instructions that will round off a number to 1 decimal place.	Follow the problem-solving steps. See **create**, **develop.**

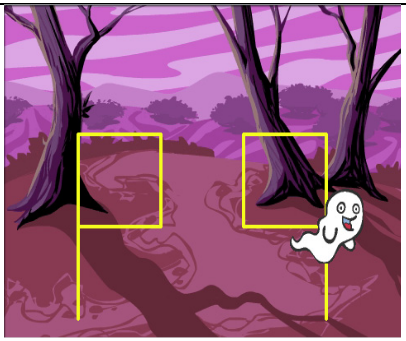
Annexure B Scratch Papers

Grade 10 June Examination	Marks:	[100]	Time:	3 hours

Instructions:

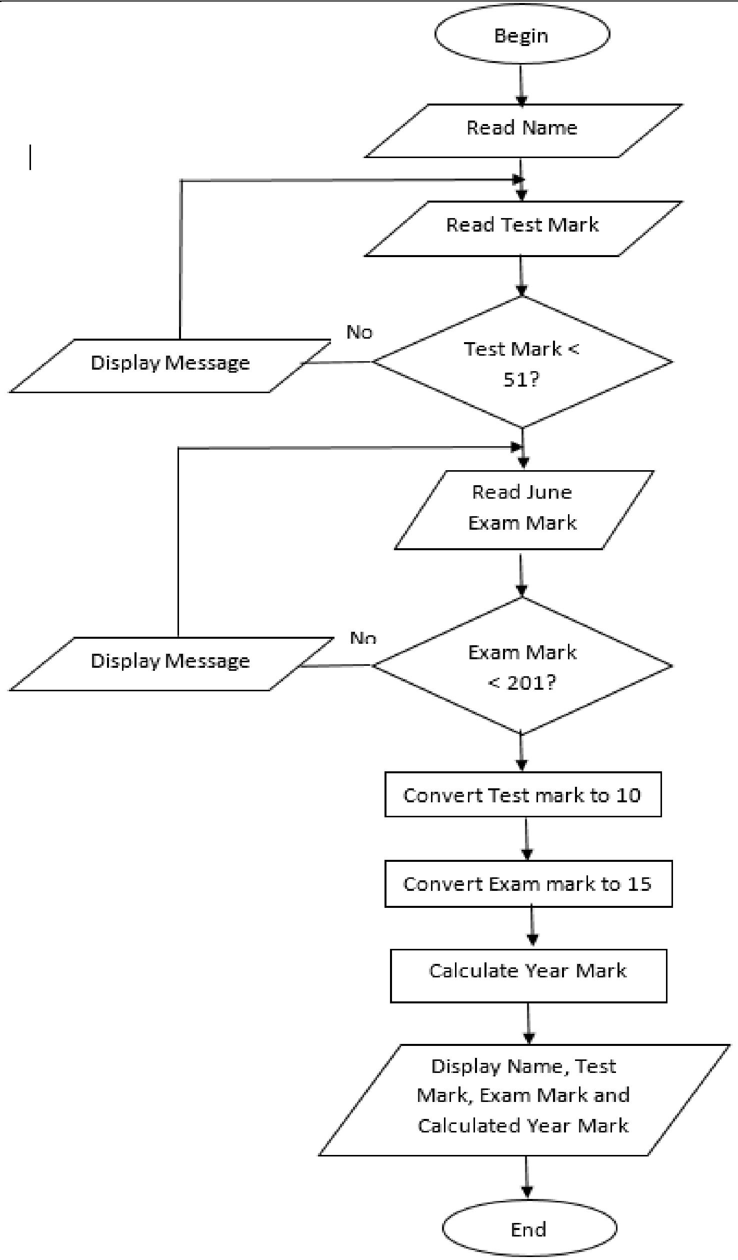
1. The duration of this examination is THREE hours. Due to the nature of this examination you will not be allowed to leave the room before the end of the session.
2. Store your projects with frequent intervals as a precaution against power failures.
3. Store your projects in a folder named Gr10**_June_XXX.** Name the projects **Question1_XXX**, **Question2_XXX** and **Question3_XXX:**
    - Replace **XXX** with your name as in the following example:
    - **Folder: Gr10_June_JohnSmith** with projects **Question1_JohnSmith**, etc.
4. Read all the questions carefully and do only what the question requires.
5. At the end of the session make sure that all your projects are stored in the prescribed folder and that all of them can be opened to be executed.
6. Each question is a separate Scratch project. Use only the names as prescribed above.

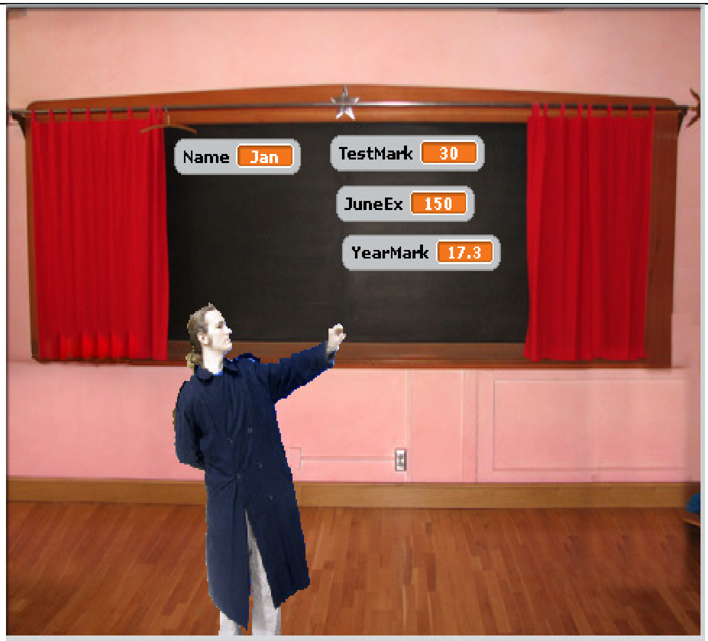
Question 1	Open the program **Question_1 LD** on your data CD Question_1 LD.sb	**[26]**	**Screenshot**	**Cognitive Demand/Comments**
1.1	Design the scene as shown according to the following instructions:			*Knowledge, easy* (C1/D1)
1.1.1	Change the background to “woods”.	(1)		Learner is able to code each instruction through recalling the structure and syntax of a concept/ structure/algorithm/ process Explicit, direct instructions – no understanding or interpretation required Coding for each instruction can be done in isolation
1.1.2	Change the sprite to “ghost-2a”.	(1)		
1.1.3	Clear the screen.	(1)		
1.1.4	Move the sprite to the following coordinates: x = -150; y = -160	(2)		
1.1.5	Let the sprite say the following for 2 seconds: “I am starting”.	(2)		
1.1.6	The size of the sprite must shrink with 50%.	(1)		

1.2	Change the appearance of the scene as shown in the screenshot by following the instructions 1.2.1 to 1.2.3:			**Cognitive Demand/Comments**
1.2.1	Change the pen colour to yellow and size to 4.	(3)		*Knowledge, easy* Recall how to change pen colour and size and how to start drawing
1.2.2	Move the sprite 200 steps up to draw the longer line. To create the square, the sprite repeats the following steps: • turn 900 and • move 100 steps forward.	(6)		*Knowledge and Understanding* Routine procedures through direct, explicit instructions that can be done in isolation (move and draw) The learner also needs to understand direction as well as interpret and understand that a loop is required to draw the square, what goes into the loop and what comes before or after the loop, etc.
1.2.3	Move the sprite and draw the reflection of the transformation. Hint: Use question 1.2.2 to draw the reflection.	(9)		*Understanding* Needs to understand how to continue from previous drawing (pen, move, direction). Needs to understand this square is a reflection of the first one and repeats the necessary code in a different position

Question 2	Open the Scratch program **Question_2 LD** on your data CD and complete the coding for the different Sprites, where applicable Question_2 LD.sb	**[15]**	Example of a short/restricted response question	**Cognitive Demand/Comments**
2.1	When Sprite2.1 is clicked, display the circumference of the circle if the diameter = 4 and Pi = 3.14 The circumference of a circle is calculated by multiplying Pi with the diameter: Pi x diameter	(2)	Recall doing a calculation in Scratch	2.1 – 2.5: *Knowledge* Recall of specific and ***isolatable*** bits of syntax/structure/processes *Use bits of code/code structures* ***in isolation*** *– no real connections (in an unrelated way)* Recall of content and skills covered in curriculum in ***exactly the same context*** *as a classroom-based exercise* Instructions are explicit, direct, to the point (no inference required – learner is able to code each line of code/concept/structure/ algorithm/process through recalling structure and syntax Context the same as classroom-based exercises
2.2.	When Sprite2.2 is clicked, write the code to determine if one number is smaller than another number. Display the message "Smaller" if Number1 is smaller than Number2 or a message "Larger" if not. **Note:** ***The variables, Number1 and Number2 are each provided with a value when the program is executed.***	(4)	Recall an algorithm (process knowledge)	
2.3	When Sprite2.3 is clicked, complete the Repeat-loop that will play the drum ***open high hat*** for **0.5** beats until the user presses the key ***d***	(4)	Recall working of repeat and drum instruction	
2.4	When Sprite 2.4 is clicked, display the rounded value of 1.75	(2)	Recall working of random instruction	
2.5	When Sprite 2.5 is clicked, display the robot's name and surname as follows: **W Masters**	(3)	Recall working of letter and join methods	

Question 3	Open the Scratch program, **Question3_LD**, on your data CD. Question_3 LD.sb	**[15]**	**Cognitive Demand/Comments**
Use the flow chart provided below and convert it to a Scratch program that will do the following: A learner's year mark (a mark out of 25) for Gr 10 IT is determined by one test and the June examination as follows: • The test mark is a mark out of 50, e.g. 36 and will count 10 out of 25 of the year mark. • A June examination is a mark out of 200, e.g. 120 and will count 15 out of 25 of the year mark. • To keep the program simple, only one sprite is used to handle all the input and output. **Note:** ***A user should not be able to enter a mark higher than the maximum mark indicated.*** If a mark exceeds the maximum mark (total for the test or examination), a message must be displayed and the user must be given the opportunity to re-enter the mark.			*Understanding (mostly)* Translating an algorithm from one form (flow chart) of representation to another form (code) Learner needs to interpret the flow chart and convert to Scratch code
Flow Chart:	**Input:** • Name of learner • Test mark out of 50 • June examination mark out of 200 **Output** • Name of learner • Test mark out of 50 • June examination mark out of 200 • Learner's year mark out of 25 At the end of the program the scene must look as follows:		Example of a structured question with a stem (scenario) and where a flow chart is provided



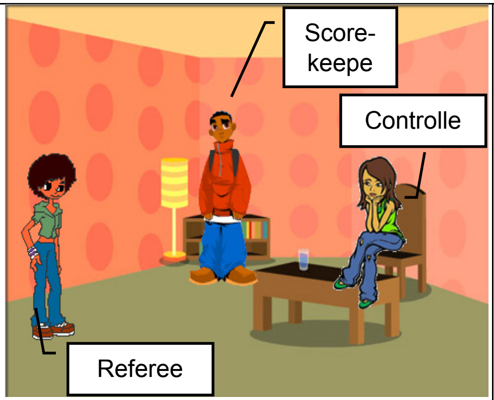


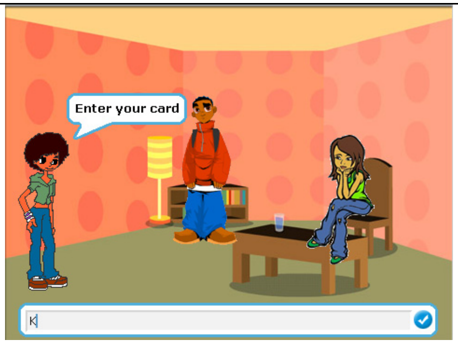
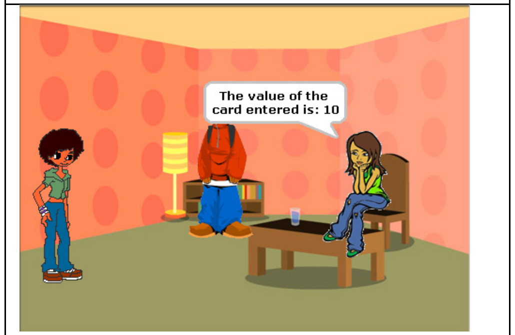
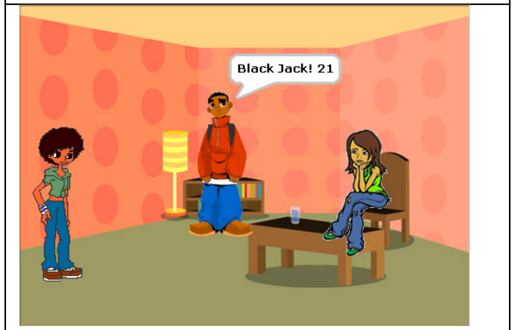
Name Jan

TestMark 30

JuneEx 150

YearMark 17.3

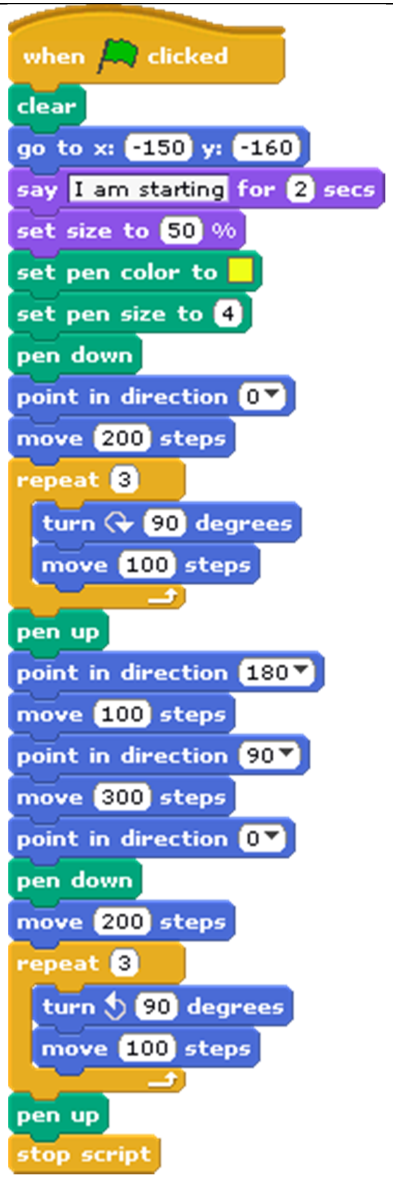
<table>
<tr><td><b>Question 4</b></td><td colspan="2">Open the program <b>Question4_LD</b> on your data CD. Question4_LD.sb</td><td><b>[32]</b></td><td><b>Cognitive Demand/Comment</b></td></tr>
<tr><td colspan="5">Blackjack is a card game in which the players attempt to get a score that is as close as possible to the value of 21, without exceeding it. In this Blackjack game a player is dealt only two cards, however, depending on the value of the first two cards dealt, the player may ask for one additional card. The player will therefore have either 2 or 3 cards. If the total value of the player’s cards exceeds 21, the player is said to be “busted.” The values of the cards are as follows:
• 10 for a Jack, Queen or King
• 1 for an Ace (A)
• Face value for all other cards (2 – 10)
Write a Scratch project that will enter the code for the player’s cards and calculate the total score of the player’s cards.</td></tr>
<tr><td colspan="2"><b>Input:</b>
The code for the player’s cards:
• A <i>capital</i> <b>J</b> for Jack, a <i>capital</i> <b>Q</b> for Queen, a <i>capital</i> <b>K</b> for King; <i>capital</i> <b>A</b> for Ace.
• The face value of the other cards (numbers 2 – 10)
<b>Note: <i>If the player does not use a third card, 0 (zero)</i> is entered</b>.</td><td colspan="3"><b>Output:</b>
• The total value of the player’s cards.
• A message, depending on the total value of the cards:
○ If the total score is exactly 21, display the message “Black Jack”
○ If the total score is less than 21, display the total score
○ If the total score is more than 21, display the message “Busted”</td></tr>
<tr><td colspan="2">The scene has three sprites:
• Referee
• Controller
• Scorekeeper</td><td colspan="2">
 </td><td><i>Application and Analysis</i>
<b>Familiar</b> context (<i>but new circumstance)</i>
Mostly, all of the information required is immediately available to the learner from the way the problem is posed (applied), however some aspects need to be inferred through understanding of relationships</td></tr>
</table>

When the program starts (the green flag is clicked): **Note:** ***Do not click any sprites*** • The **referee** requests the player to enter the codes (one-by-one) for his/her cards. ○ A *capital* **J** for Jack, a *capital* **Q** for Queen, a *capital* **K** for King, *capital* **A** for Ace ○ The face value of the other cards (numbers 2 – 10) ○ 0 (zero) if no third card is required • Each time a card code is entered, the **controller** immediately tells the player what the value of that card is. Card values: ○ 10 for a Jack, Queen or King ○ 1 for an Ace (A) ○ Face value for all other cards (2 – 10) • After the player entered his/her third card (either the code or a 0), the **scorekeeper** communicates a message (see **output** above) depending on the total value of the player's cards.	 Enter your card K  The value of the card entered is: 10  Black Jack! 21	*Application and Analysis* *Application* Learner needs to identify appropriate structures and combine programming building blocks (without being explicitly told what and where) to complete the task *Analysis* Requires ***reasoning***, ***developing*** a plan and sequence of steps Needs to understand how the parts (3 sprites and their actions) relate to each other and how the data will flow between the different parts (broadcast and *when* to broadcast *what*) Example of a structured question with some open-ended elements

Question 5	Write a Scratch program to display the ***next*** 10 numbers in a Tribonacci series with signature (0,1,1)	**[11]**	**Cognitive Demand/Comments**
The Tribonacci numbers start with three predetermined terms and each term afterwards is the sum of the preceding three terms. The Tribonacci numbers T_n are defined as follows: $T_1 = 0$, $T_2 = T_3 = 1$, and $T_n = T_{n-1} + T_{n-2} + T_{n-3}$ (n ≥ 3) In other words, each number is the sum of the previous three numbers. The first few Tribonacci numbers with signature (0,1,1) are: 0, 1, ,1, 2, 4, 7, 13, 24 … (**Note**: The first three terms are the signature. The signature may vary.)			*Analyse and Create* Free response question that includes analysis, pattern recognition, generalisation of pattern. Learner needs to relate, combine and integrate several concepts /code structures/ constructs to devise a 'new' algorithm or adapt existing ones to solve the problem Example of a free-response / open-ended question

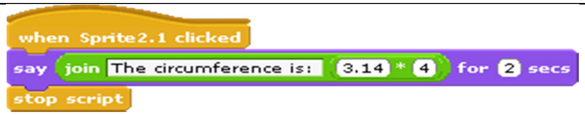
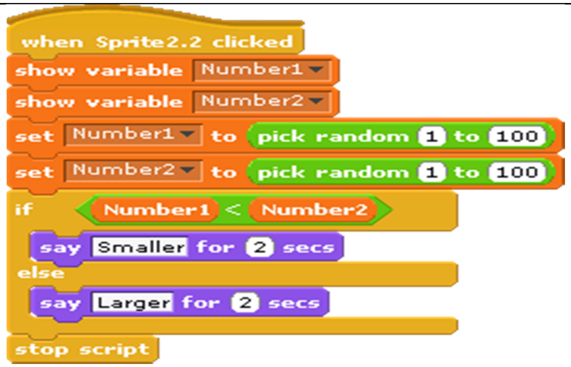
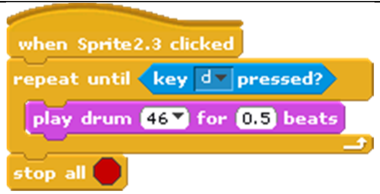
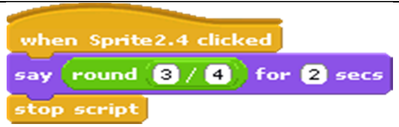
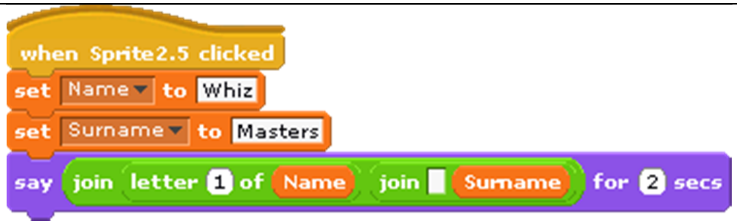
Memorandum

Question 1

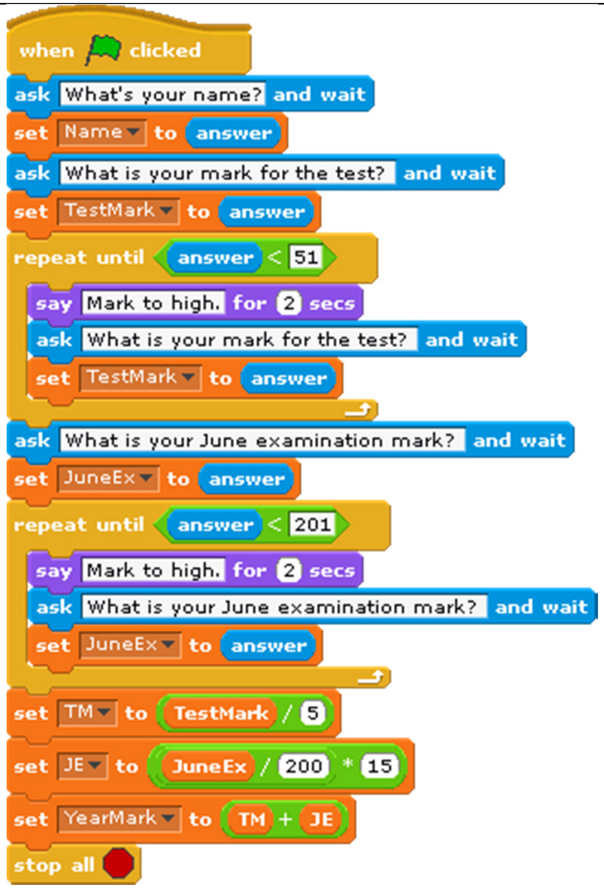
1.1.1.	Change the background to woods✓	**1**		**Solution/Comments**
1.1.2	Change the sprite to ghost-2a✓	**1**		
1.1.3	Clear the screen. ✓	**1**		
1.1.4	Pen up ✓ x coordinate -150✓ y coordinate -160✓	**3**		
1.1.5	Sprite says for 2 seconds "I am starting" ✓	**1**		
1.1.6	Set the sprite size to 50% ✓	**1**		
1.2.1	Pen down ✓ Set pen colour to yellow ✓ Set pen size to 4 ✓	**3**		
1.2.2	Point in direction 0 degrees ✓ Move 200 steps ✓ (before repeat) ✓ Repeat 3 times ✓ Move sprite 100 steps ✓ Turn 90 degrees ✓	**6**		
1.2.3	Pen up ✓ Point in direction 180 degrees ✓ Move 100 steps ✓ Point in direction 0 degrees ✓ Pen down ✓ Duplicate repeat steps in 1.2.2 ✓ different turn✓ Move 200 steps (before repeat) Repeat 3 times Turn 90 degrees Move 100 steps ✓ Repeat steps in 1.2.2 Pen up ✓	**9**		
		26		

```
when 🏴 clicked
clear
go to x: -150 y: -160
say I am starting for 2 secs
set size to 50 %
set pen color to 🟨
set pen size to 4
pen down
point in direction 0▾
move 200 steps
repeat 3
    turn ↻ 90 degrees
    move 100 steps
pen up
point in direction 180▾
move 100 steps
point in direction 90▾
move 300 steps
point in direction 0▾
pen down
move 200 steps
repeat 3
    turn ↺ 90 degrees
    move 100 steps
pen up
stop script
```

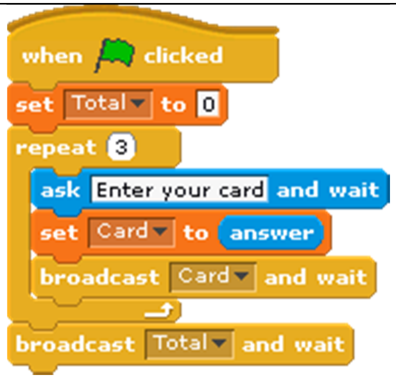
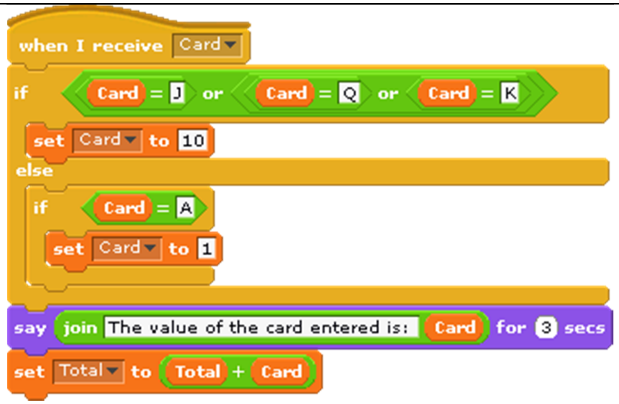
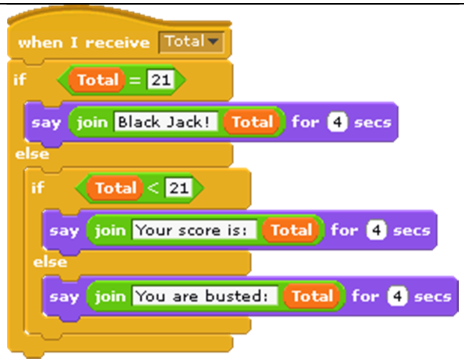
Question 2

2.1	Correct calculation 3.142 ✓ X 4 ✓	**2**	 when Sprite2.1 clicked say join The circumference is: (3.14 * 4) for 2 secs stop script	If learners only calculated the circumference (without displaying it), mark is allocated
2.2.	If ✓ Number1 < Number2 (correct comparison) ✓ else display message✓ then display message✓	**4**	 when Sprite2.2 clicked show variable Number1 show variable Number2 set Number1 to pick random 1 to 100 set Number2 to pick random 1 to 100 if Number1 < Number2 say Smaller for 2 secs else say Larger for 2 secs stop script	**Note:** Learners were only required to code the if-statement (the first part was hard coded (given))
2.3	Repeat loop – Correct condition ✓ Play drum instruction within loop ✓ Correct drum (46) ✓ for 0.5 beats ✓	**4**	 when Sprite2.3 clicked repeat until key d pressed? play drum 46 for 0.5 beats stop all	**Note** The repeat loop was provided. Learners only had to complete the condition and insert the instruction
2.4	Correct method ✓and correct values ✓	**2**	 when Sprite2.4 clicked say round (3 / 4) for 2 secs stop script	If learners only rounded the value (without displaying it), marks are allocated
2.5	Extract first letter of name ✓ Join first letter of name with space ✓ and surname ✓	**3**	 when Sprite2.5 clicked set Name to Whiz set Surname to Masters say join letter 1 of Name join Surname for 2 secs	**Note** The name and surname are hard coded If initial and surname are not displayed, marks are allocated
		15		

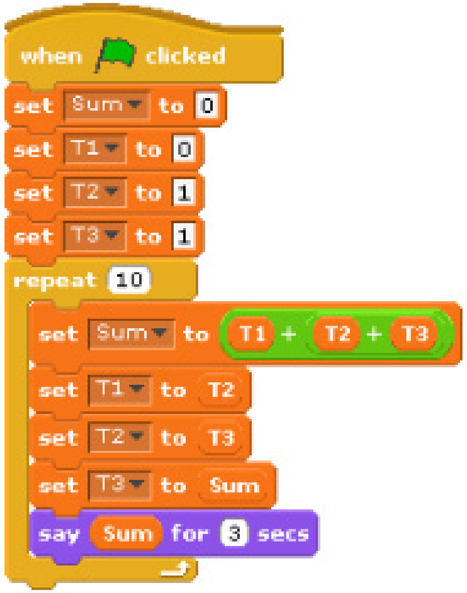
Question 3

	Min 4 variables used: ✓Name; Test Mark; June Exam; Year Mark **Input** Input the Name and Test Mark ✓ Repeat ✓ until answer <= 50 ✓ Message: Mark is too high ✓ Input Test Mark again ✓ Do the same for June Exam mark ✓ until answer < 200 ✓ **Processing** Convert Test Mark to 10: TestMark/5 ✓✓ Convert Exam Mark to 15: ExamMark/200*15 ✓✓ Calculate Year Mark: TestMark+ExamMark ✓ **Output** Display to output ✓ according to the scene ✓		 when [flag] clicked ask What's your name? and wait set Name to answer ask What is your mark for the test? and wait set TestMark to answer repeat until answer < 51 say Mark to high. for 2 secs ask What is your mark for the test? and wait set TestMark to answer ask What is your June examination mark? and wait set JuneEx to answer repeat until answer < 201 say Mark to high. for 2 secs ask What is your June examination mark? and wait set JuneEx to answer set TM to TestMark / 5 set JE to JuneEx / 200 * 15 set YearMark to TM + JE stop all	Here we see a bit of a repetition of the same aspects, i.e. testing the same skill (entering and testing the two different marks as well as conversion of the marks) Therefore, only two marks are allocated for the second repeat loop (entering the Exam Mark) The totals for the two marks were therefore deliberately chosen with the idea that different formulas could be used **Testing the same skills in a paper or test, should be avoided**, however, real-life scenarios often force this, in which instance the mark allocation must address it (though one could have been provided and the other one asked). Repetition of skills can also be addressed through hard coding (however take care that answers for previous or subsequent questions are not given away through this)
		15		

Question 4

	Variables created✓ min 2 ✓ **Referee Script: (Input)** Set Total value to 0 ✓ correct place (before loop) ✓ Repeat ✓ 3X ✓ Ask to enter card code ✓ Set card code to code entered ✓ Broadcast card code to Controller ✓ inside loop✓ Broadcast Total value to Scorekeeper ✓ outside loop✓	**12**	 when 🏁 clicked set Total to 0 repeat 3 ask Enter your card and wait set Card to answer broadcast Card and wait broadcast Total and wait	Another approach could have been followed, however broadcast is required for the way the question is posed. This was done deliberately to increase the cognitive demand of the question.
	Controller Script: When I receive card code ✓✓ **Test card codes** If ✓J, Q or K ✓✓set value to 10 ✓ else if ✓A ✓set value to 1 ✓ Display card value ✓ Add card values ✓✓	**12**	 when I receive Card if Card = J or Card = Q or Card = K set Card to 10 else if Card = A set Card to 1 say join The value of the card entered is: Card for 3 secs set Total to Total + Card	The repetition of the if-statement could be seen as repeating the same skill, however the contexts are different and forms part of a bigger solution (unlike in Question 3 where the first one could be used as is (copied), only changing variable/values) Separate if-statements could have been used – re-allocate marks to accommodate it.
	Scorekeeper Script (Output) When I receive total value✓✓ **Test Total** If total value = 21 ✓, display message *Black Jack* ✓ else if ✓total value < 21✓ display message with score ✓ else ✓ display message *Busted* with score ✓	**9**	 when I receive Total if Total = 21 say join Black Jack! Total for 4 secs else if Total < 21 say join Your score is: Total for 4 secs else say join You are busted: Total for 4 secs	
		33		

Question 5

	Outside loop: Correct statements✓ Set sum to 0 ✓ Set T1, T2 and T3 to 0,1,1 (signature) respectively✓ Repeat 10 ✓times Set sum to adding ✓ T1, T2 and T3 ✓ Set T1 to T2 ✓ Set T2 to T3 ✓ Set T3 to Sum ✓ Display Sum ✓ (Correct statements in loop ✓)		 when clicked set Sum to 0 set T1 to 0 set T2 to 1 set T3 to 1 repeat 10 set Sum to T1 + T2 + T3 set T1 to T2 set T2 to T3 set T3 to Sum say Sum for 3 secs	
		11		

Task Description	Comments
This following task is a Grade 10 Practical November examination that covers the practical content and skills in the Grade 10 curriculum: • Recall of basic code constructs and methods • Interpreting pseudo code and converting to code • Application of code constructs • Analysis • Problem solving	The questions are scaffolded and start with easy and moderately difficult cognitive level 1 (knowledge) questions (Q1 – Q2 and part of Q7). Question 1 tests knowledge Question 2 tests knowledge (including process knowledge and algorithms) Question 3 tests understanding Question 4 tests application Question 5 tests application Question 6 tests basic reasoning, therefore easy to moderately difficult analysis Question 7 tests knowledge (prime algorithm) and problem solving Also note that repetition of skills (testing a skill more than once) is avoided, however overlap of skills are encountered in some questions due to problem-solving nature. Where there is a repetition of content and skills, the contexts are different.

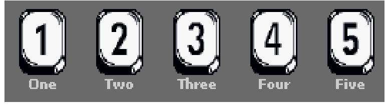
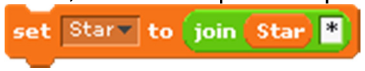
Cognitive Analysis

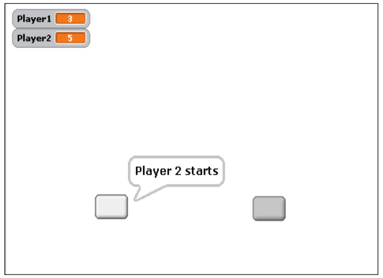
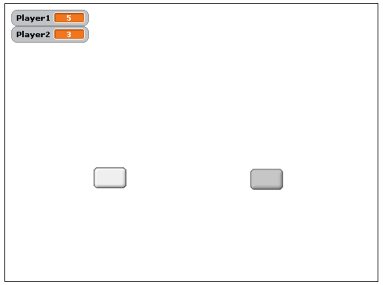
Cognitive Level	Lower Order	Middle Order	Higher Order	Total
Q1	13			**13**
Q2	9			**9**
Q3		15		**15**
Q4		22		**22**
Q5	2	3	11	**16**
Q6	12		21	**33**
Total	**36 (33%)**	**40 (37%)**	**33 (31%)**	**108**

Grade 10 November Examination	Marks:	[120]	Time:	3 hours

Instructions:

1. The duration of this examination is THREE hours. Due to the nature of this examination you will not be allowed to leave the room before the end of the session.
2. Store your projects with frequent intervals as a precaution against power failures.
3. Store your projects in a folder named Gr10**_November_XXX.** Name the projects **Question1_XXX**, **Question2_XXX** and **Question3_XXX:**
   - Replace **XXX** with your name as in the following example:
   - **Folder: Gr10_November_JohnSmith** with projects **Question1_JohnSmith**, etc.
4. Read all the questions carefully and do only what the question requires.
5. At the end of the session make sure that all your projects are stored in the prescribed folder and that all of them can be opened to be executed.
6. Each question is a separate Scratch project. Use only the names as prescribed above.

Question 1	10 T4 Q1 LD.sb Open the program **10 T4 Q1 LD** on your data CD.	**[13]**	**Screenshot**	**Cognitive Demand/Comments**
1.1	Write code for sprite **One** that will calculate and display the square root of 625 when the ***Green Flag*** is clicked.	2	The program has five sprites:  1 2 3 4 5 One Two Three Four Five You must code each sprite separately – they are not related.	*Knowledge, with different levels of difficulty*
1.2	When sprite **Two** is clicked, write code that will do the following: • Create a variable called, *Position*, that only sprite Two will have access to and set the value to 3 • Insert the number *007* into position3 of the list, Code.	4		Learner is able to code each instruction through recalling the structure and syntax of a concept/ structure/algorithm/ process Explicit, direct instructions – no understanding or interpretation required Coding for each instruction can be done in isolation
1.3	When sprite **Three** is clicked, write a Repeat loop that will execute the instruction,  set Star to join Star * 5 times to create 5 stars (*****) as one string	3		
1.4	When sprite **Four** is clicked, write code that will determine how many characters the string, *Heibo! Brother*, consists of.	2		
1.5	When sprite **Five** is clicked, write code that will change the value of *Number* with 2 times the value of X	2		

Question 2	Open the Scratch program, **10 T4 Q2 LD**, on your data CD. 10 T4 Q2 LD.sb	[9]	**Learner data /Screenshot**	**Cognitive Demand/Comments**
A simple dice program simulates the throw of a dice. When the program starts (Green flag is clicked), each of two players is assigned a number from 1 to 6. The number for Player 1 is assigned to the variable, Player1 and the number for Player 2 is assigned to the variable, Player2 (***You do not need to code this as this part is already coded and provided***). The player with the biggest number 'wins' and starts the game. ***Note: Assume that players will not be assigned the same number.*** 2.2 Complete the code for the Start sprite (when sprite Start is clicked) that will display which player (Player 1 or Player 2) will start the game: • Write an if-statement that will determine which player has the highest number and display which player will start the game: *Player 1 starts* if Player 1 has the highest number or *Player 2 starts* if Player 2 has the highest number 2.3 Sometimes players prefer to swap numbers before the 'winner' is determined. When the *Swap* sprite is clicked, complete the code that will swap the numbers of the two players.			**Start-button:**  Player1 3 Player2 5 Player 2 starts **SWAP-button:**  Player1 5 Player2 3	*Knowledge* Recall of algorithm for determining the biggest/smallest one of two numbers (2.1) Recall of the algorithm that swaps two values (2.2) The code for each sprite can be coded in isolation (and there is no synchronisation required between the two Sprites)
Input: Input is provided (hard coded)	**Output**: 2.1 Message that indicates which player starts 2.2 No output required			

Question 3	Open the Scratch program, **10 T4 Q3 LD**, on your data CD.	**[15]**	**Cognitive Demand/Comments**

Compound interest

10 T4 Q3 LD.sb

When you borrow money, you pay interest. If you borrow money for a period longer than one year, the interest of the previous year is added to the original loan amount and the interest for the subsequent year is calculated on the "new", bigger amount. So, the interest amount you pay grows annually as each year you pay "interest on interest".

This is called compound interest.

The table below illustrates how compound interest is calculated over a five-year period when borrowing R1 000.00 at 10% annual interest rate and how much you will owe after five years:

Year	Loan at Start	Interest Amount	Loan at End
0 (Now)	R1 000.00	(R1 000.00 × 10% =) **100.00**	R1 100.00
1	R1 100.00	(R1 100.00 × 10% =) **110.00**	R1 210.00
2	R1 210.00	(R1 210.00 × 10% =) **121.00**	R1 331.00
3	R1 331.00	(R1 331.00 × 10% =) **133.10**	R1 464.10
4	R1 464.10	(R1 464.10 × 10% =) **146.41**	R1 610.51
5	R1 610.51		

Use the pseudo code below and convert it to a Scratch program that will calculate the annual loan at start, interest amount and loan at end as well as the final amount that needs to be paid back.

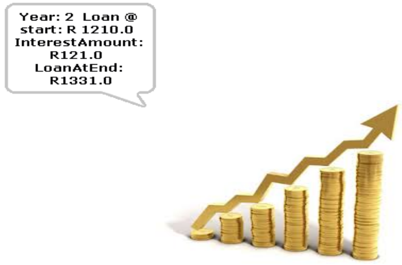
Understanding (Easy)

Translating an algorithm from one form (pseudo code) of representation to another form (Scratch code)

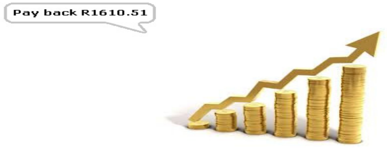
Learner needs to interpret the pseudo code and convert it to Scratch code

Pseudo Code:

```
LoanAtStart ← 1000
Year ← 0
Interest Rate ← 10%
Interest ← 0
Repeat for Number of Years
        Interest ← 10% of LoanAtStart
        LoanAtEnd ← Sum of LoanAtStart and Interest
        Display Year, LoanAtStart, Interest and LoanAtEnd
        LoanAtStart ← LoanAtEnd
        Year ← Year + 1
Display Amount to be paid back
```



```
Year: 2 Loan @
start: R 1210.0
InterestAmount:
R121.0
LoanAtEnd:
R1331.0
```



```
Pay back R1610.51
```

Input:

Loan amount
Number of years
Interest rate

Output

Final amount to be paid back

Question 4

Open the Scratch program, **10 T4 Q5 LD**, on your data CD.

10 T4 Q5 LD.sb

[21]

Cognitive Demand/Comments

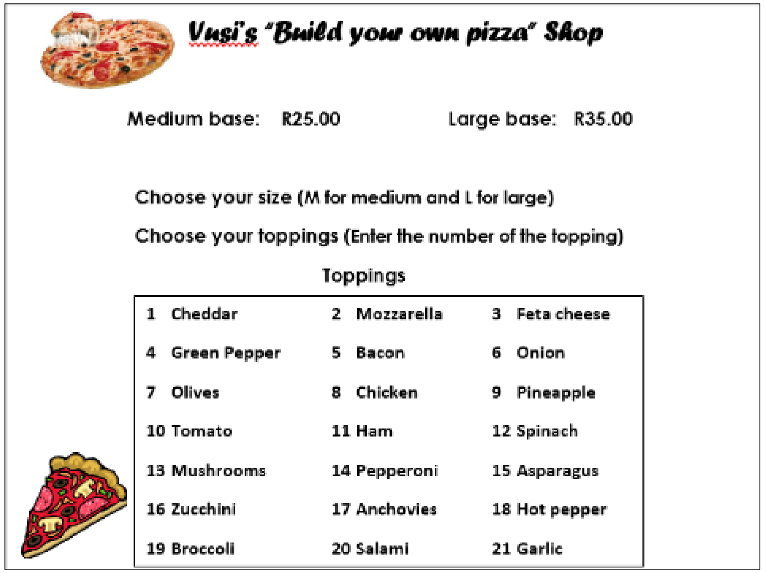
At Vusi’s pizza shop one builds one’s own pizza.

They sell a base pizza at a fixed price and the customer then adds his/her own toppings, any number and in any combination, from a list of toppings. The price of each topping is then added to the ‘base’ price to determine the final price of the pizza.

A pizza is available in two sizes: Medium and Large.

The different topping prices for a medium pizza are stored in a list, Medium, and the different topping prices for a large pizza are stored in the list, Large.

Note:
The prices are stored in exactly the same order as the toppings, e.g. the price of olives (topping no. 7) is stored in position number 7 on both price lists.



Vusi's "Build your own pizza" Shop

Medium base: R25.00 Large base: R35.00

Choose your size (M for medium and L for large)

Choose your toppings (Enter the number of the topping)

Toppings

1 Cheddar	2 Mozzarella	3 Feta cheese
4 Green Pepper	5 Bacon	6 Onion
7 Olives	8 Chicken	9 Pineapple
10 Tomato	11 Ham	12 Spinach
13 Mushrooms	14 Pepperoni	15 Asparagus
16 Zucchini	17 Anchovies	18 Hot pepper
19 Broccoli	20 Salami	21 Garlic

The user enters M (medium) or L (large), depending on the size of the pizza ordered. Then the user enters the numbers (1 – 21) of the toppings that must be added to the pizza, one at a time. If there are no more toppings to be added, the number 0 is entered.

The base price for the medium pizza is R25.00 and for the large pizza it is R35.00. Thereafter the price of each chosen topping is added to the base price to make up the final price of the pizza.

Write a Scratch program that will allow the user to choose between a Medium and Large pizza and then choose all the toppings that must be added. Finally, display the price of the pizza ordered.

Application (Difficult)

Learners need to apply their knowledge and understanding of lists to look up topping prices and to calculate the price of the pizza.

All information required to complete the task is immediately available to the learner.

The **process** (working of lists) ***is known*** to the learner and is applied to a problem that is ***familiar***, but that has not been solved previously in the same context or with the same data or with the same tools (**new** circumstance or **new** elements added)

It requires application of appropriate concepts/processes/ routines/ constructs/algorithms ***without having to be prompted***…and ***without having to be shown how to use it in a familiar context***

Input:	**Output**
Size of pizza (M or L) Number(s) of the topping(s) to be added, e.g. 5 for bacon 0 (zero) if no more toppings are required.	Price of the pizza

Question 5

10 T4 Q6 LD.sb

Open the Scratch program, **10 T4 Q6 LD**, on your data CD
Write a Scratch program that finds the numeric key of an encrypted message.

[16]

Cognitive Demand/Comments

To encrypt a message, the algorithm uses a randomly generated *key* consisting of a *single positive integer* k (1 ≤ k ≤9).

It takes as input a plaintext string consisting of *uppercase* letters. It then produces as output a *ciphertext* string consisting of *uppercase* letters, i.e. the plaintext string contains no lower case letters and no special characters such as full stop, space, etc.

The algorithm works in the following way: Converts each character of the plaintext string into a number, such that ‘A’ = 0, ‘B’ = 1, ‘C’ = 2, ... ‘Z’ = 25.

1. For each number n, compute a new number y, such that y is the *remainder* of $\mathbf{n^2}$ multiplied by **k**, divided by **26**.
2. Convert each new number back into a character, such that 0 = ‘A’, 1 = ‘B’, 2 = ‘C’, ... 25 = ‘Z’.
3. Add the corresponding letter for the key as follows:
    a. If the length of the encrypted message is even, add the key to be in the middle position of the message
    b. If the length of the encrypted message is uneven, add the key to be in the second last position of the message.

For instance, suppose the input is the string “HELLO”, and the key, k = 5 .

1. The characters in the string are converted to numbers to yield [7, 4, 11, 11, 14].
2. Take the square of each number, multiply by 5, then take the remainder when divided by 26. This yields the numbers [11, 2, 7, 7, 18].
3. Convert each number to a character to yield “LCHHS”.
4. Add the key, F (5 → F) to yield “LCHHFS”

Your job is to use the ciphertext and find the numeric value of the key that will eventually enable one to decrypt the message.

The following are sample inputs with corresponding outputs.

Analysis

Although a very clear explanation of how the encryption works is provided, which could imply application (level 2), the clear description with examples is needed to understand the problem in order to complete the task. However, the solution to the problem requires a degree of reasoning and analysis.

The learner needs to figure out that, originally, an even string becomes an uneven string when the key is added in the middle position. Similarly, an uneven string becomes even, therefore the key would be in the second last position.

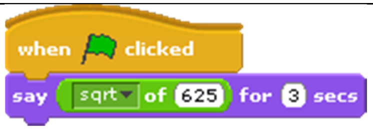
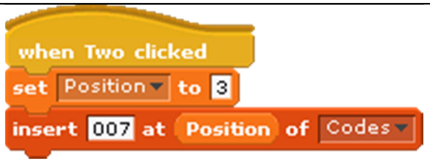
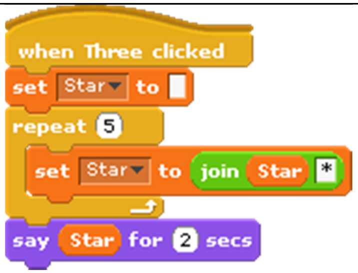
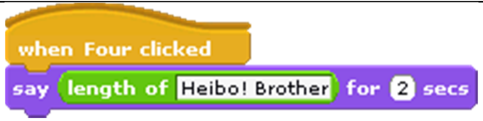
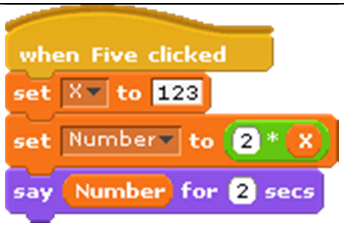
Requires reasoning, developing a plan or sequence of steps (algorithm); has some complexity.

Input:		**Output**:	
Cipherstring		*Numeric value of the key*	
Input 1:	LCHHFS	Output 1:	5
Input 2:	SSUIIAG	Output 2:	8

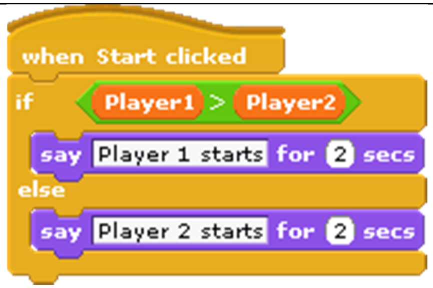
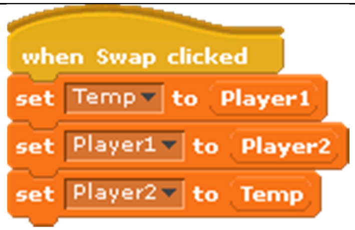
Question 6	Test if the remaining numbers in a prime number is prime when the original prime number is truncated 10 T4 Q7 LD.sb	**[33]**	**Cognitive Demand/Comments**
Truncatable prime numbers: The prime number 37973 has an interesting property. It is possible to continuously remove digits from left to right, and each time a digit is removed, the remaining part is still a prime number: 3797, 797, 97, and 7. Similarly, if one removes digits from the right to left, the remaining part is still a prime number: 3797, 379, 37, and 3. **Note:** ***Single-digit prime numbers 2, 3, 5, and 7 are not considered to be truncatable primes.*** There are two sprites, *Digit* and *Prime*. *Prime* ***only*** tests if a number is prime or not. Write a Scratch program that will do the following: • Read a prime number from the list, *PrimeNumbers* ○ Remove one digit at a time from the left and test if the remaining number is prime or not ○ Add the prime number ”truncs” (remaining parts) to the list, *Truncs*.			*Knowledge and Analysis* The way the question is posed, the code for sprite Prime tests *knowledge/recall* of a known algorithm (to determine whether a number is prime or not). This part can be coded in isolation (separate sprite, *Prime*), regardless of whether a learner is able to succeed with the code for the other sprite (Digit) or not. To write the code for sprite, Digit, requires seeing the relationship between parts of a whole, reasoning, generalisation of a pattern, analysis required.
Input: *Prime number from the list, PrimeNumbers* Input 1: 37973 Input 2:	**Output** *A list with all the prime ‘truncs’ obtained from the prime number* Output 1: 3797, 797, 97, 7 Output 2:		Synchronising the two sprites, requires understanding of data flow between parts (*analysis*)

Memorandum

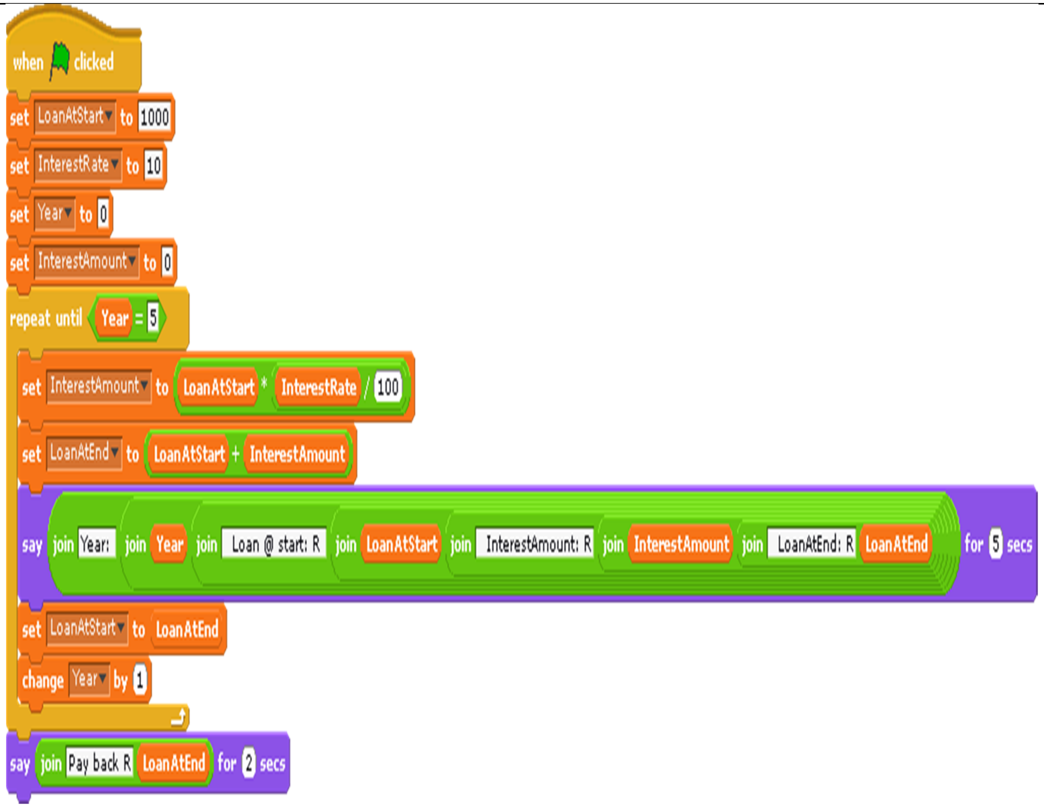
Question 1

1.1	Determine sqrt of 625 ✓ Display sqrt of 625 ✓	**2**	 when ⚑ clicked say sqrt of 625 for 3 secs	**Note:** Learners needed to calculate and display the sqrt
1.2.	Position declared as local variable (this sprite only) ✓ Position set to 3 ✓ Insert 007 ✓ in correct position of list (Position) ✓	**4**	 when Two clicked set Position to 3 insert 007 at Position of Codes	**Note:**
1.3	Repeat used ✓ correct number of times (5) ✓ Provided instruction placed inside loop ✓	**3**	 when Three clicked set Star to repeat 5 set Star to join Star * say Star for 2 secs	**Note** The repeat loop was provided. Learners only had to complete the condition and insert the instruction
1.4	Length of string calculated ✓ and displayed ✓	**2**	 when Four clicked say length of Heibo! Brother for 2 secs	
1.5	Number set to 2 x ✓ X ✓	**2**	 when Five clicked set X to 123 set Number to 2 * X say Number for 2 secs	**Note** X is provided Number variable is created
		13		

Question 2

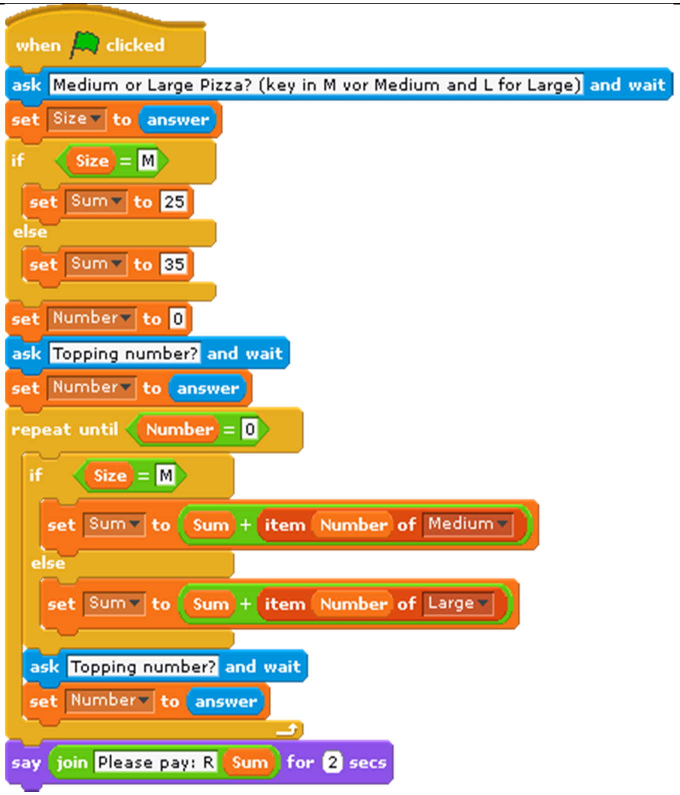
3.1	If ✓correct condition ✓ then display winner ✓ else ✓ display other winner ✓	**5**	 when Start clicked if Player1 > Player2 say Player 1 starts for 2 secs else say Player 2 starts for 2 secs	
3.2	Declare Temp ✓ Set Temp to Player1 ✓ Set Player 1 to Player 2 ✓ Set Player 2 to Temp ✓	**4**	 when Swap clicked set Temp to Player1 set Player1 to Player2 set Player2 to Temp	**Note:**
		9		

Question 3

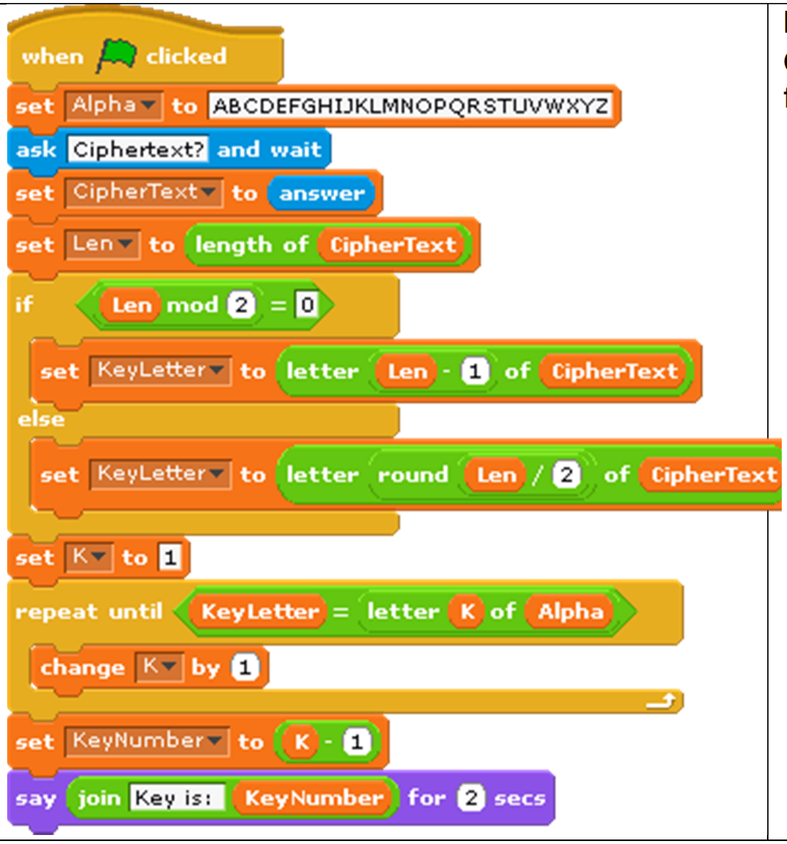
	Set LoanAmount to 1000 ✓ Set InterestRate to 10 ✓ (*The above could also have been entered via keyboard or any other method*) Set Year to 0 ✓ Set InterestAmount to 0 ✓ Repeat ✓ correct condition ✓ Calculate interest ✓ Calculate loan @ end ✓ Display in loop ✓ all ✓ correct ✓ Change loan @ start to loan @ end ✓ Increase year by 1 ✓ Correct statements inside and outside loop ✓ Display payback amount ✓			
		15		

```
when [flag] clicked
set LoanAtStart to 1000
set InterestRate to 10
set Year to 0
set InterestAmount to 0
repeat until <(Year) = 5>
    set InterestAmount to ((LoanAtStart) * ((InterestRate) / 100))
    set LoanAtEnd to ((LoanAtStart) + (InterestAmount))
    say (join Year: (join (Year) (join Loan @ start: R (join (LoanAtStart) (join InterestAmount: R (join (InterestAmount) (join LoanAtEnd: R (LoanAtEnd)))))))) for 5 secs
    set LoanAtStart to (LoanAtEnd)
    change Year by 1
say (join Pay back R (LoanAtEnd)) for 2 secs
```

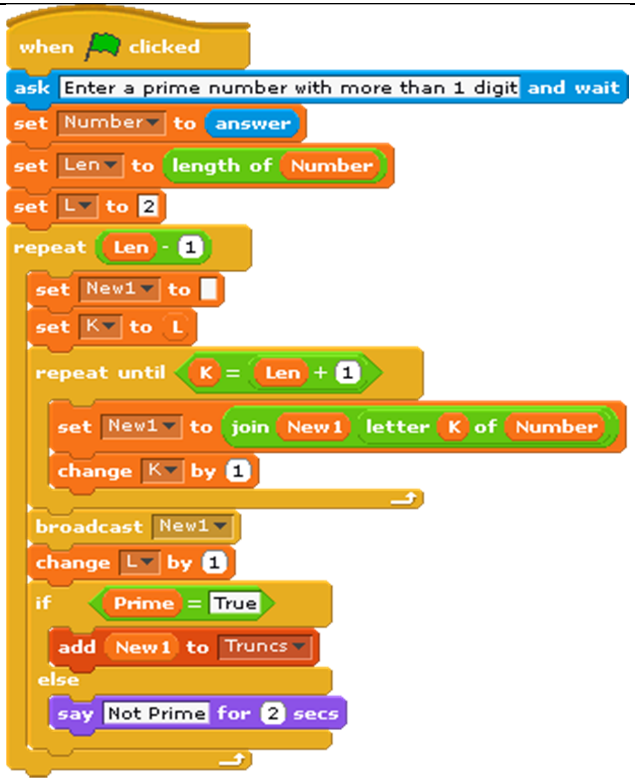
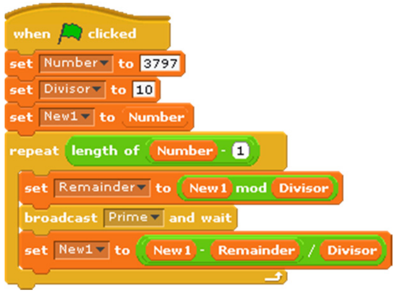
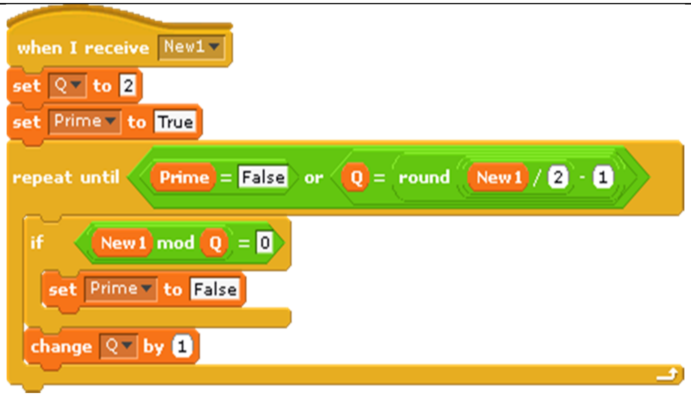
Question 4

	Create min three variables (size, sum, number) ✓ Enter size ✓ If ✓size = M ✓then Sum = 25 ✓ else ✓ Sum = 35 ✓ Initialise number to 0 ✓ Enter first topping Number ✓ *Correct position – before list* ✓ Repeat ✓ until Number = 0 ✓ If ✓ Size = M ✓then set Sum to Sum ✓+ Price (Item Number) ✓from Medium list ✓ else ✓ set Sum to Sum + Price (Item Number)✓ from Large list ✓ Enter topping Number ✓ Display pizza price ✓		 when 🏴 clicked ask Medium or Large Pizza? (key in M vor Medium and L for Large) and wait set Size to answer if Size = M set Sum to 25 else set Sum to 35 set Number to 0 ask Topping number? and wait set Number to answer repeat until Number = 0 if Size = M set Sum to Sum + item Number of Medium else set Sum to Sum + item Number of Large ask Topping number? and wait set Number to answer say join Please pay: R Sum for 2 secs	
22				

Question 5

	Enter Ciphertext ✓ Determine length of Ciphertext ✓ **Find key:** If ✓length is even ✓, get key in second last position: Letter[Length–1] ✓of Ciphertext else ✓ get key for uneven length (middle of Ciphertext) in position: round ✓ (length/2) ✓ Set K (alphabet index) to 1 ✓ before loop ✓ **Find key number:** Repeat ✓until Key letter✓ = Alphabet letter ✓ Set K (alphabet index) to point to next alphabet letter ✓ Set Key number to K-1 ✓ after loop ✓ Display Key number (not necessary)		 when 🏁 clicked set Alpha to ABCDEFGHIJKLMNOPQRSTUVWXYZ ask Ciphertext? and wait set CipherText to answer set Len to length of CipherText if Len mod 2 = 0 set KeyLetter to letter Len - 1 of CipherText else set KeyLetter to letter round Len / 2 of CipherText set K to 1 repeat until KeyLetter = letter K of Alpha change K by 1 set KeyNumber to K - 1 say join Key is: KeyNumber for 2 secs	**Note:** Other approaches can be followed.
		16		

Question 6

Truncs	Enter Prime Number Determine length of Prime Number ✓ Initialise outside loop ✓counter: Set L to 2 ✓ **Outer loop:** Repeat✓ length-1 ✓times Initialise remaining part (New1) to null string ✓ Set letter index (K) to L ✓ **Inner loop:** Repeat ✓until letter index (K) is length+1✓ ***Build remaining part (New1):*** Join New1✓and Letter[K] of Prime Number✓ Increase letter index by 1 ✓ **After inner loop:** Broadcast New1 ✓ (to test if prime) Increase L by 1 ✓ If ✓remaining part (New1) is prime ✓ Add New1 to Prime list ✓ else ✓display message "not prime" ✓ Correct instructions within outer and inner loops ✓	**20**	 when 🏁 clicked ask Enter a prime number with more than 1 digit and wait set Number to answer set Len to length of Number set L to 2 repeat Len - 1 set New1 to set K to L repeat until K = Len + 1 set New1 to join New1 letter K of Number change K by 1 broadcast New1 change L by 1 if Prime = True add New1 to Truncs else say Not Prime for 2 secs	**Note:** "Truncating" the number can also be done using mod together with round, instead of string manipulation.  when 🏁 clicked set Number to 3797 set Divisor to 10 set New1 to Number repeat length of Number - 1 set Remainder to New1 mod Divisor broadcast Prime and wait set New1 to New1 - Remainder / Divisor String manipulation is often more demanding in Scratch than in Delphi due to more methods available in Delphi.
Prime	**Code for sprite, Prime:** When I receive ✓ Set quotient (Q) to 2 ✓ Set Prime to true ✓ Repeat ✓ until Prime is false ✓or ✓ Q = (round(New1)/2)-1 ✓ If ✓ New1 mod Q = 0 (test if Q is factor) ✓ Set Prime to false ✓ Change Q by 1 ✓ Correct instructions inside loop ✓	**12**	 when I receive New1 set Q to 2 set Prime to True repeat until Prime = False or Q = round New1 / 2 - 1 if New1 mod Q = 0 set Prime to False change Q by 1	**Note:** Using a different sprite to determine whether the number is prime, helps to separate this part from the 'main' program and will allow a learner to earn these marks (which is a known algorithm – recall of process knowledge) even if he/she could not get the first part right.
		32		

222 Struben Street, Pretoria, 0001
Private Bag X895, Pretoria, 0001, South Africa
Tel: 012 357 3000 • Fax: 012 323 0601

Private Bag X9035, Cape Town, 8000, South Africa
Tel: 021 486 7000 • Fax: 021 461 8110
Call Centre: 0800 202 933

ISBN 978-1-4315-3124-0

Department of Basic Education

www.education.gov.za

www.twitter.com/dbe_sa

www.facebook.com/BasicEd