education

Department:
Education
REPUBLIC OF SOUTH AFRICA

INFORMATION TECHNOLOGY

GUIDELINES FOR PRACTICAL ASSESSMENT TASK

2010

These guidelines consist 29 pages.

Practical Assessment Task (PAT)

Information Technology

2010

Guidelines to the teacher

A Introduction

The objective of the Practical Assessment Task is to allow the teacher to directly and systematically observe and evaluate the applied competence of the learner. The PAT comprises the application of the knowledge (content, concepts and skills) to Information Technology.
In Information Technology the PAT counts 25% of the total promotion/ certification mark for the subject (i.e. 100 marks out of 400 marks). It is implemented across the first three terms of the school year and should be undertaken as one extended task, which is broken down into different phases or a series of smaller activities.

The IT PAT comprises of four components:

- Investigation and Analysis
- Design
- Coding and Implementation
- Documentation and General evaluation

B Programming Project

1. What is the programming project?

The programming project of the Practical Assessment Task in Information Technology requires the learner to develop a software solution for a particular problem within a given scenario.

2. Planning for the programming project

2.1 Description

Scenario for the 2010 Grade 12 PAT:

The Department of Basic Education has launched a competition for the Grade 12 IT learners to develop educational software (software designed to facilitate teaching and learning) to be used to teach or learn a concept or a skill in South African schools.
For the purpose of this competition the educational software must be aimed at any concept or content of the eight learning areas for Grades R – 9 or any of the 29 recognised subjects for Grades 10 – 12.
You may choose your own topic or focus area within the context of the given scenario.
Examples of possible types of educational software to choose from:

- Drill and practice exercises such as for spelling or mathematical concepts
- Simulating a science experiment
- Testing content knowledge
- etc.

The learner must identify a topic in one of the learning areas or subjects and develop a software solution that teachers could use to teach certain concepts or content or that learners could use to learn, practise or drill certain content or concepts. The software could also encourage competitions, based on subject content, between learners or schools.

The program must interact with a database and use a GUI to interact with the user. Either Java or Delphi can be used as a programming language. In addition to the program, you must also compile a technical manual and a user guide for the program.
The learner may choose his/her own topic/application within the context of the given scenario.

In completing the project the learner will apply the following skills

- Investigation
- Analysis
- Design
- Software development
  - Programming skills using the programming language studied
  - Database development
  - Graphical User Interface (GUI) design

Sections of the assessment tool will penalise projects that are not related to the given scenario.

2.2 Development phases of the project

The programming project will be completed in 4 phases indicated in the following table.

Phase	Marks	%
Phase 1: Investigation and Analysis	33	16.5
Phase 2: Design	30	15
Phase 3: Coding and Implementation	100	50
Phase 4: Documentation and General evaluation	37	17.5
Total	**200**	**100**

IMPORTANT: Documentation/evidence of what the learner did during each phase of development must be submitted at specified intervals. It is vital that evidence is supplied for all phases of work. The required document for each of these phases is given in the learner section of the document.

Deadlines for handing in the final product of each phase will be set by the teacher taking into account the moderation dates for the different phases. The product of each phase will be assessed and the marks will be recorded.

2.3 Requirements for the project

The learners should adhere to the following minimum criteria:

Investigation and Analysis

- Description of the problem in his/her own words outlining the main aspects in one paragraph.
- Investigate the topic. The investigation will consist of information obtained from the real world situation in which the end software product is to be used.
- Preliminary investigation to identify the nature and scope of the problem and to gather facts.
- Analysis of the problem – What are the requirements and what should the programming solution provide?
  (See Learner section 2, Phase 1, and Assessment tool Phase 1)

Design

- Design a solution – how will the program/system meet the requirements? Provide a well-planned solution to the problem.
  (See Learner section 2, Phase 2, and Assessment tool Phase 2)

Coding and Implementation

- The project must include the major development tools, i.e. database design and programming in an integrated manner. (Other applications could be integrated with these development tools)

- Other aspects of the programming project that will be assessed include:
  - Programming style
  - Graphical User Interface (GUI)
  - Use of Human-Computer Interaction (HCI) principles
  - Expertise required and functionality of the program
  - Robustness of the program including the use of defensive programming techniques
  - Whether the project matches its original aims and goals

  (See Learner section 2 Phase 3, Assessment tool Phase 3)

Documentation and general evaluation

- Document the solution, installation procedures and hardware and software requirements – Technical Manual
- Compile a user guide
- Evaluate the following:
  - Time management of the learner – Did he/she meet all the deadlines?
  - Appropriateness of the solution in the context of the scenario.

  (See Learner section 2 Phase 4, Assessment tool Phase 4)

3. Instructions to the learners

See Learner section of this document.

4. Resources

The learner will need the following resources to complete the project:

- Access to a computer with the following programs:
  - Programming language: Java or Delphi
  - Word processor such as MS Word
  - Database software such as MS Access
- IDE (for Delphi it is part of the programming language but for Java you will need additional software such as JBuilder/Turbo JBuilder/Netbeans/Eclipse/JCreator)

5. Assessment of the PAT

The task should be completed under controlled conditions and facilitated and continuously monitored by the teacher.

See Assessment Tool Section for the assessment sheets for the different phases.

Teachers must ensure that learners receive the following documents at the beginning of their Grade 12 year:

- The “Instructions to the learner“-section included in this document
- The assessment sheets for all phases included in this document

Learners should be allowed to reflect on the marks they have obtained and address mistakes they have made before completing the next phase, BUT the marks allocated for the initial evaluation will NOT be revised.

Learners will be required to demonstrate their system for debriefing at the end of phase 3. Teachers should evaluate the projects according to the assessment tool provided for phase 3.

Learners will **NOT** be allowed to change the topics of their projects once phase 1 and 2 have been completed and assessed. If a learner should decide to change his/her topic after phase 1 or phase 2 has been assessed, the learner has to redo phase 1 and/or phase 2 for the new topic. In this case the teacher will NOT re-assess the updated phase 1 and/ or 2. The marks for the original phases must be recorded. However, phase 3 will not be assessed unless phase 1 and 2 reflects investigation that was done for the new topic.
Correlation between all the phases of a project should be strictly and continuously checked during assessment. Evidence of work done during previous phases must always be available during each assessment and moderation of a specific phase of development.

Guidelines for the demonstration and internal evaluation of the project:

- The teacher must schedule dates and times for demonstrations. Allow approximately 15 minutes per project for the demonstration as well as about 5 minutes for setting up the project and getting feedback from the teacher afterwards.
- The development of the project is a continuous process. The teacher should always look at the work that has been done in the previous phases when assessing a specific phase of development. The teacher should monitor the progress of the project closely in relation to the work that was done during the previous phases.
  - The requirements identified and set out in phase 1 should be reflected in phase 2 - the design phase of the project.
  - The work done during the design phase – phase 2 - should be reflected in coding and implementation of the project – phase 3.
  - The documentation done during phase 4 should correspond with the coded project – phase 3.
- The learner should have all previous documentation (phase 1 and phase 2) handy when the demonstration of phase 3 takes place.
- The demonstrations must be done electronically on the computer.
- The learner must execute his/her computer program and show all the features of the program to the teacher for assessment.
- The teacher must ask the learner to perform test strategies to test all the facets of the program.
- The teacher can require of the learner to execute other additional test procedures to make sure that the entire program is working correctly.
- The teacher must use the mark sheet for phase 3 to allocate marks during the demonstration.
- The teacher must **identify random pieces of programming code (excluding the 10% borrowed code) in the project. The learner must then explain the purpose and working of the randomly selected code to the teacher**. This is done to ensure that learners do the coding themselves. A similar type of procedure will be followed during the external moderation. If the learner cannot explain the code used in the project, **no marks can be awarded to the learner for the project.**
- The learner must hand in the electronic copy of the project that was demonstrated. The teacher will use this copy to allocate any outstanding marks in order to finalise the mark.

NOTE: Once the product of a phase has been handed in and assessed that phase will not be re-assessed.

6. Recording and Reporting

For each phase the teacher will assess the phase, record the mark and give feedback to the learner.
The marks for the different phases are added and converted to a mark out of 100 which will be the final mark.

7. Guidelines for managing the PAT

There are two ways to go about managing the programming project:

Option 1:

- The teacher could dedicate one or two periods per week to the PAT while continuing with normal teaching to complete the Grade 12 curriculum in the rest of the week.
- If he/she chooses this option, he/she should start with the PAT towards the end of the first term, completing one phase per term.

Option 2:

- The teacher could dedicate a continuous period of time to the PAT, e.g. the last week of each term, completing one phase per term.

The teacher will have to plan his/her work schedule according to the option that he/she prefers.
It is suggested that the teacher "registers" the learners' topics when they start with phase 1 to avoid "instant projects" that might occur and could possibly not be the learner's own work. Teachers should also ensure that learners "register" projects which they are capable of completing so as to prevent topic changes due to the initial topic being too difficult or time consuming.

8. HINTS

- Before learners start with Phase 1, first explain the PAT and provide an overview of the process of development to the learners.
- Discuss the task and the topic with the learners. Allow them to ask questions and ensure that they clearly understand the problem to be solved.
- Discuss examples of possible topics within the given scenario with the learners. Let the learners come up with some ideas and discuss the appropriateness thereof.
- Although a different scenario was used, it might be useful to let learners see both good and bad examples of PATs from previous years.
- Be very strict with the handing in of the phases.

INFORMATION TECHNOLOGY

2010

PRACTICAL ASSESSMENT TASK (PAT)

INSTRUCTIONS TO THE LEARNER

INFORMATION TECHNOLOGY – PRACTICAL ASSESSMENT TASK (PAT)

The IT PAT comprises of four components:

- Investigation and Analysis
- Design
- Implementation and Coding
- Documentation and general evaluation

1. Planning for the programming project

1.1 Description

Scenario for the 2010 Grade 12 PAT:

The Department of Basic Education has launched a competition for the Grade 12 IT learners to develop educational software (software designed to facilitate teaching and learning) to be used to teach a concept or a skill in South African schools.

For the purpose of this competition the educational software must be aimed at any concept or content of the eight learning areas for Grades R – 9 or any of the 29 recognised subjects for Grades 10 – 12.

You may choose your own topic or focus area within the context of the given scenario.

Examples of possible educational programs to choose from:

- Drill and practice exercises such as for spelling or mathematical concepts
- Simulating a science experiment
- Testing content knowledge
- etc.

Identify an area in one of the learning areas or subjects and develop a software solution that teachers could use to teach certain concepts or content or that learners could use to learn, practise or drill certain content or concepts. The software could also encourage competitions based on subject content between learners or schools.

The program must interact with a database and use a GUI to interact with the user. Either Java or Delphi can be used as a programming language. In addition to the program, you must also compile a technical manual and a user guide for the program.

In completing the project you will apply the following skills:

- Investigation
- Analysis
- Design
- Software development
  - Programming skills using the programming language studied
  - Database development
  - Graphical User Interface (GUI) design

Sections of the assessment tool will penalise projects which are not related to the given scenario.

Note: Your final program must comprise of one single logically related piece of software. Projects which consist of two or more unrelated programs will only obtain marks for one of the parts since only one of the programs will regarded as the actual project.

1.2 Development phases of the project

The programming project will be completed in 4 phases indicated in the following table:

Phase	Marks	%
Phase 1: Investigation and Analysis	33	16.5
Phase 2: Design	30	15
Phase 3: Coding and Implementation	100	50
Phase 4: Documentation and general evaluation	37	17.5
Total	**200**	**100**

Documentation/evidence of what you did during each phase of development must be submitted at specified intervals. The evidence and output for each of the phases is discussed below. Marks can only be awarded if this evidence is supplied to your teacher AND if the work to be assessed is in relation to what has been done during previous phases.

Dates for submitting the documentation/evidence will be set by the teacher.

Study the assessment tools beforehand to make sure that you have addressed all the relevant requirements according to the assessment tools.

Consider the feedback from the teacher indicated on the assessment tools and improve your work for the next phase accordingly. In a number of instances marks are awarded for correcting work done incorrectly in previous phases.

All the documentation of the previous phases must be available to the teacher during each assessment.

1.3 Resources required for the project

You will need the following resources to do the project:

- Access to a computer with the following programs:
  - Programming language: Java or Delphi
  - Word processing such as Word
  - Database software such as Access
- IDE (for Delphi it is part of the programming language but for Java you will need additional software such as JBuilder/Turbo JBuilder/Netbeans/Eclipse/JCreator)

The project must be completed under **controlled conditions** and facilitated and continuously monitored by the teacher.

You need to adhere to the following minimum criteria:

Investigation and Analysis

- Description of the problem in your own words outlining the main aspects in one paragraph.
- Investigate the topic: The investigation will consist of information obtained from real world situations and scenarios where the software will be used.
- Preliminary investigation to identify the nature and scope of the problem and to gather facts from potential users including their needs and any limitations they may have.
- Analysis of the problem – What are the requirements and what should the programming solution provide?

(See the following section Phase 1 and the Assessment tool Phase 1)

Design

- Design a solution – how will the program/system meet the requirements? Provide a well-planned solution in terms of:
    - Input, processing and output
    - Structure and contents of the database
    - GUI and flow of events.

(See the following section Phase 2 and the Assessment tool Phase 2)

Coding and Implementation

- The project must include the major development tools, i.e. database design and programming in an integrated manner. (Other applications could be integrated with these development tools)
- Other aspects of the programming project that will be assessed include:
    - Programming style
    - Graphical User Interface (GUI)
    - Use of Human-Computer Interaction (HCI) principles
    - Level of expert programming
    - Functionality of the program
    - Robustness of the program including the use of defensive programming techniques
    - Whether the project matches its original aims and goals

(See the following section Phase 3 and the Assessment tool Phase 3)

Documentation and general evaluation

- Printout of source code including comments, installation procedures and hardware and software requirements – Technical Manual
- Compile a user guide
- Demonstration and debriefing of final product.
- The teacher will evaluate the following
    - Your time management – Did you meet all the deadlines?
    - Appropriateness of the solution in the context of the scenario.

(See the following section Phase 4 and the Assessment tool Phase 4)

2. Instructions for the phases of the programming project

The instructions for the different phases are as follows:

PHASE 1:

Investigation and Analysis **Due date:**_____

In completing this phase you need to find some background information on your topic and determine *what* the program/system should do and provide:

1. Problem Statement
   - Identify and describe/explain in your own words the task and the problem to be solved. This description should not be a description of any computer code or the solution. All that is required is a description of the problem that you are investigating in the real world context or situation where it has been identified.
   - Your statement should
     - clearly state what the problem entails,
     - outline the aspects that should be solved and
     - indicate what the purpose of the software will be.

     Example:

     I am going to design software to be used by Grade 3 learners to practise and drill addition and subtraction as part of their numeracy learning programme. Currently this is done manually and the manual system relies on the teacher to check answers and give feedback to learners. This is time-consuming and also makes it difficult for the teacher to pay attention to many learners during class time. The advantage of the software will be that it will test the concepts, give immediate feedback to learners and provide them with the correct answers and explanations and will let them proceed with more advanced exercises as soon as they have mastered a concept. The teacher will have time on hand to pay attention to learners that struggle and need personal attention or further explanation. The software will also provide the look and feel of a game, which learners will enjoy. The software will be able to check the learners' answers for correctness, provide immediate feedback, keep records and provide reports of the learners' scores, achievements and progress as well as recommend to the learner when to move on to more advanced exercises.
2. Investigation
   - Investigate the topic to gather some facts and to determine the nature and scope of the problem, e.g. what will be part of the software and what it will not include. The aim is to determine requirements, needs and limitations.
   - You might want to find out the following:
     - Which teachers or schools might use the type of software you are developing
     - What systems or processes are used in the absence of software (e.g. paper-based solutions)
     - Information specific to your topic (e.g. if you are developing a program for Grade 3 learners you need to know what the range is for adding and subtracting numbers, whether they work with negative numbers, etc)
   - Demographics of users of your system (age, gender, disabilities, literacy level, etc)
   - You must keep a comprehensive reference list of all resources (websites, books) used.
   - Evidence of investigation: Well formatted and presented summary of all the findings

3. Determine requirements for the program
   - Conduct discussions with end-users, make notes, gather documents and compile summaries and draw conclusions from this information.
   - Observe how the system or a similar system currently works. If possible, make notes.
   - Make notes on input, processing and output that will be required.
   - Possible evidence to be submitted: Questionnaires, notes (or recordings) made during interviews, photographs, documents/forms from current system, audio/video from interviews/how the system currently works etc.
   - Information must be supplied on details of the interviews, questionnaires, users etc. Information such as the time and date the questionnaire/interview was conducted as well as the names, location and contact details of participants. Your teacher may use this information to verify that the investigation has been done in the way indicated by the evidence that you have provided.
4. Possible solution
   - Using your investigation and responses from users write a possible/intended brief solution for the problem in your own words.
   - Possible evidence to be submitted: Problem definition and/or requirements list. This is not a detailed specification with input and output but rather a simple list of features that the intended solution will provide to users.
5. Submit a planning document
   Combine the results of your investigation in a planning document consisting of:
   - Description of the problem
   - Background information on the topic with references
   - Evidence of methods used to gather information as well as details of users or participants.
   - Broad overview of a possible solution to the problem

PHASE 2:

Design **Due date:_____**

In completing this phase you need to determine the specifications for the program/system and *how* the program/system will accomplish the goals set during analysis. **Study the assessment tool for Phase 2 and 3 to make sure that your project meets the requirements that all tasks must adhere to.**

1. Convert requirements/features into specifications
   Specify the solution in terms of:
   - Input
   - Processing
   - Output (tables, graphs, currency, units of measurement, etc included)

   Include the following as part of the specifications
   - Defensive programming techniques such as:
     - Data validation (indicate where and how it should be applied)
     - Error messages
   - Data structures to organise and store data
     - Database design
       - fields (types, names and size)
       - relationships
       - keys (primary and foreign)
       - ER diagrams
     - Data types and structures used for programming :
       - Primitive types – single value, e.g. integer, real/double
       - Structured data types – collections of data, e.g. arrays, matrices, records, sets, combo boxes, lists, vectors.
       - Abstract data types – collections with set of data and set of operations that can be performed on data, e.g. classes and objects
       - Design of classes – methods with parameters and return data types, attributes, constructors, class diagrams, inheritance hierarchies if applicable (higher order).
   - Graphical User Interface (GUI) design
     Drawings and/or screenshots of GUI indicating the following:
     - Layout of components
     - Navigation (e.g. how does user get from one screen to another and back)
     - Flow of events (from one screen to another, from one event to another)
     - Data flow between units/modules (what data is passed between modules?)
     - Justification for use of input and output components (e.g. why was a combo box used instead of a text field, or why was a checkbox used instead of a radio button).
2. Submit a document with all the program specifications as listed above. The document must include the following:
   - Input, processing and output requirements
   - Error checking and validation procedures
   - Database Design
   - List of Data Structures used (including class diagrams)
   - Design and layout of the GUI

The document could include IPO tables, flow charts, and diagrams such as ER diagrams, class diagrams and/or screen dumps that will give a clear representation of the system including required notes well as descriptions of specifications.

PHASE 3:
Coding and Implementation **Due date:** _____

In completing this phase you will have to code the solution including the GUI as planned in the previous phase, create the data structures, debug and test the program. **Study the assessment tool for Phase 3 to make sure that your project meets the requirements that all tasks must adhere to.**

Suggested steps to complete this phase:

1. Break up the solution as outlined in phase 2 into modules (e.g. according to options).
2. Code/Create the GUI.
3. Create the data structures to organise and store data.
4. Coding – Code the solution according to requirements and specifications (input, processing and output) compiled in the previous phase. Among other things marks will be awarded for the following:
   - Appropriate input strategies
   - Database access
   - Appropriate and effective algorithms
   - Good programming principles: code re-use, variable names, commenting
   - Parameter passing: Independence of modules
5. Error handling – Ensure that input is validated where required and that exception handling is applied where required.
6. Testing/Debugging – Ensure that the program runs correctly and is error free
   - Apply a test strategy to ensure that the program/system does what it should by using different ranges of data including extreme/erroneous test data. Keep track of this data as it will be needed for your documentation.
7. Evaluate your program/system – Does the program do what it should/meet the requirements? Did you apply good programming principles?
   - Review program code. Have good programming principles been applied? Study the assessment tool for phase 3 intensively.
   - Does the program/system operate properly?
   - Does the program comply with what was stated in Phase 1 and 2?
   - Well-designed GUI?

   Note: You are allowed to make use of borrowed code but it cannot exceed more than 10% of your programming code. This will typically be code to perform unusual functions such as playing a video clip etc. **Your program will not be accepted by the teacher if it exceeds the limit of 10% of borrowed code**.

PHASE 4:

Documentation and Evaluation **Due date: _____**

In completing this phase, you will have to finalise the documentation pertaining to the solution:

1. **Technical Documentation**

The technical documentation/manual must include the following:

- General description of the problem and a broad solution as compiled in phase 1.
- Database design. Screenshots of the design view of the fields and their data types of the different tables as well as the relationships between the tables or a description thereof.
- Printouts of the code for unusual system functions used, for example, use of system clock/CD player.
- Specifications of test procedures and test results and error recovery procedures.
- Relationships/communication between modules (i.e. show what screens/modules communicate with one another and what data is passed between them).
- Sample runs of the program with results: A printout of the results showing that typical data has been used. The program must also be tested with extreme or erroneous data.
- Source code of all modules

2. **User Guide**

The user guide should include the following:

- Title sheet and table of contents.
- Background to the project. Here the history of the development of the project should be described.
- How to use the software – detail depends on the complexity of the software and the user-friendliness of the interface and help functions.
- Scope and limitations of the program
- Hardware required to run the program (hard drive space, memory, CPU, special input devices, etc.)
- Software required in addition to the program itself, for example, proprietary classes, operating system, etc
- Installation instructions.
- Files that are used and their contents. The format/layout of each file must be included.
- Detailed instructions for the user and a walk-through of the program.
- Input required. The exact format of the input should be specified, particularly if formatted input is being used, for example a date.
- Output/reports produced.
- Troubleshooting for potential problems
- Future developments. Given more resources, e.g. time and software, what additional features could be implemented?
- References and acknowledgements, especially where third party software is used.

3. **Hand in:**

- Electronic copy of program and all applicable files e.g. data files, etc.
- Technical Manual
- User Guide

4. Demonstrate the program for evaluation and debriefing
Guidelines for the demonstration of the project:

- The teacher will schedule dates and times for demonstrations. About 20 minutes per project will be allowed.
- The learner should hand in all the documentation before the demonstration takes place – at least one week in advance.
- The demonstrations must be done electronically on the computer.
- The learner must execute his/her computer program and show all the features of the program to the teacher for evaluation.
- The teacher can use a test strategy provided in the technical manual as a guideline and ask the learner to perform parts of or all the test strategy.
- The teacher can require of the learner to execute other additional test procedures to make sure that the entire program is working correctly.
- The teacher can use the mark sheet for Phase 3 as a guideline and allocate marks accordingly during the demonstration.
- As part of the demonstration, the teacher will **identify random pieces of programming code (excluding the 10% borrowed code) in the project. The learner must then explain the purpose and working of the randomly selected code to the teacher**. This is done to ensure that learners did the coding themselves. A similar type of procedure will be followed during the external moderation. If the learner cannot explain the code used in the project, no marks can be awarded to the learner for the project.
- The learner must hand in the electronic copy of the project that was demonstrated. The teacher will use this copy to allocate any outstanding marks in order to finalise the mark.

5. Final general evaluation

The teacher will evaluate the following:

- Time management – did you meet all the deadlines?
- Utility value – Is your solution appropriate in the context of the given scenario?
- Do the different phases of development correlate and lead to the final solution as a continuous process?

Information Technology

2010

Practical Assessment Task (PAT)

Assessment Tools

Assessment tools for the programming project

Assessment for Phase 1 **Name of learner:**_____

Investigation and Analysis: Criteria						Possible Mark	Mark Obtained
Problem statement	**The problem statement**					**4**	
	4	**3**	**2**	**1**	**0**		
	The problem is clearly stated and described and unambiguous – clearly states what the problem entails. Outlines the aspects that should be solved. Clear statement of what the purpose of the software will be	The statement is clear but with minor shortcomings.	The statement is vague, leaving the reader unsure of what the purpose of the system will be.	The statement is so vague that no discernable purpose can be found.	No statement or description		
Investigation	**Key areas pertaining to the topic**					**4**	
	4	**3**	**2**	**1**	**0**		
	Investigation clearly and comprehensively defines/explains key areas pertaining to the topic. Shows good insight and understanding into all key areas of the topic	Minor shortcomings Shows insight in most of the key areas of the topic	Shows some insight in some of the key areas	Vague Shows little understanding of key areas	Key areas not defined		
	List of references					**2**	
	2		**1**		**0**		
	Comprehensive list In acceptable format		Less comprehensive – minor shortcomings		No references		
Requirements	**Input requirements**					**3**	
	3	**2**	**1**	**0**			
	Comprehensive list of input requirements obtained from users, well defined and obtained using at least two different methods such as questionnaire, interview.	Less comprehensive list of input requirements obtained from users using at least one method. Not that clearly defined but still acceptable	Few input requirements obtained from users or requirements made up by learner. Vague and not clearly defined.	No input requirements obtained			

	Processing requirements					
	3	**2**	**1**	**0**		
	Comprehensive list of processing requirements obtained from end-user, well defined and obtained using at least two different methods	Less comprehensive list of processing requirements obtained from users using at least one method. Not that clearly defined but still acceptable	Few processing requirements obtained from users or requirements made up by learner. Vague and not clearly defined.	No processing requirements obtained	**3**	
	Output requirements					
	3	**2**	**1**	**0**		
	Comprehensive list of output requirements obtained from users, well defined and obtained using at least two different methods such as questionnaire, interview,	Less comprehensive list of output requirements obtained from users using at least one method. Not that clearly defined but still acceptable	Few output requirements obtained from users or requirements made up by learner. Vague and not clearly defined.	No output requirements obtained	**3**	
	Methods					
	2		**1**	**0**		
	Used at least 2 different methods to obtain requirements, e.g. questionnaires, sample documents, notes/audio/video on interviews/investigation, etc.		Used only 1 method to obtain requirements, e.g. questionnaires, sample documents, notes/audio/video on interviews/investigation, etc.	No requirements obtained	**2**	
	Evidence of collecting information regarding requirements e.g. questionnaires, sample documents, notes/audio/video on interviews/investigation, etc. (1 mark for each type/method used for collecting relevant evidence up to a maximum of 3 marks)				**3**	
Possible solution	**Possible solution:**					
	4	**3**	**2**	**1**	**0**	
	Clear and comprehensive description – a clear overview of a possible solution is given. Clearly indicates what the software will include and do	Clear but with minor shortcomings	Basic description is given but some aspects of the suggested solution are vague.	The description is vague. No clear solution can be identified in the description.	No possible solution given	**4**
General	**The scope (boundaries and assumptions/Features that will be included)**					
	3	**2**	**1**	**0**		
	The scope of the suggested solution is clearly and well defined Clearly indicates what will be part of the program and what it will not include	Defined but some minor shortcomings. Not always clear on what the program will include and not include	Vaguely defined No clear scope can be determined.	Not defined		**3**
	Appropriateness of the suggested solution in the context of the scenario:					
	2		**1**	**0**		
	Most appropriate Good application for scenario		Appropriate Application in scenario not always convincing	Not appropriate Application in scenario not convincing		**2**
					Total:	**33**

Assessment for Phase 2: **Name of learner:** _____

	Design: Criteria				Possible Mark	Mark Obtained
Specifications	**Input**				3	
	3	**2**	**1**	**0**		
	User input and other sources of input clearly described in terms of what input is required and the format (e.g. date format) of the input	Described with minor shortcomings in terms of what is required and format.	The description of input is vague or incomplete.	Only listed – no description.		
	Processing				3	
	3	**2**	**1**	**0**		
	How data will be processed / manipulated clearly described in terms of requirements, format, calculations, formulas, etc. Short, clear and correct description of in all instances where applicable. User clear on result.	Description of processing/manipulation of data done/clear in most instances where required. Good effort, but can improve. Not always correct or applicable	Description of processing/ manipulation of data not clear in most instances. Not done in all instances. Descriptions vague/ incomplete. Not correct/applicable in most instances	Processing/ manipulation of data not described		
	Output				3	
	3	**2**	**1**	**0**		
	Identified and clearly described required output (screen, reports) according to requirements and in terms of format (currency, units of measurement, etc)	Identified and described with minor shortcomings	The description is vague or most required output not identified or format of output not described	Output not identified and described		
	Data validation				3	
	3	**2**	**1**	**0**		
	Indicated for all input and described in detail. Meaningful and effective	Identified and described with minor shortcomings	Indicated in some cases where applicable and/or description of validation vague or incomplete	Not Indicated		
	Error messages associated with data validation				2	
	2		**1**	**0**		
	Indicated for all applicable validation checks/errors and described/shown in detail.		Indicated some error messages and/or description of error messages vague	No error message indicated for any data validation or error		
Database design overview	**Fields:**				2	
	2		**1**	**0**		
	Well chosen fields, field types and sizes to suit the application. Field properties such as validation and masks indicated where applicable.		Mostly well chosen fields, field types and sizes to suit the application. Field properties such as validation and masks not always indicated where applicable.	Not well chosen fields, field types and sizes to suit the application. Field properties such as validation and masks not indicated.		

	Tables					
	3	**2**	**1**	**0**		
	Fields well grouped into tables. No repetition of fields in tables. More than two tables with correct relationships between tables. Shows correct primary and foreign keys.	Fields well grouped into tables with minor shortcomings. Some repetition of fields in tables. At least two tables with correct relationship between tables. Shows correct primary and foreign keys.	Fields mostly well grouped into tables/ extensive repetition of fields in tables. At least two tables but no/ incorrect relationship between tables. Shows no/incorrect primary and foreign keys.	Only one table/only the fields listed	**3**	
Data structures	**Organise and store data** (collections with set of data and set of operations that can be performed on data e.g. classes and objects)					
	3	**2**	**1**	**0**		
	Used abstract data types. Application well planned in terms of classes and objects. Object well-structured with relevant methods	Used abstract data types. Room for improvement. Application not entirely planned in terms of classes and objects. Not always applicable. The entire program is not object-oriented	An effort to use abstract data types. Objects not always well compiled. Objects compiled but not used correctly/not used at all. Very little of program is object-oriented	No abstract data types (classes and objects)	**3**	
GUI Design	**Input**					
	3	**2**	**1**	**0**		
	Appropriate input components to support accurate/valid input in all instances where required. Substantiated choices where required	Appropriate components where required. Choices mostly substantiated Minor shortcomings	Appropriate in most instances where required. Choices not always substantiated.	Not done/not appropriate output components identified	**3**	
	Output					
	3	**2**	**1**	**0**		
	Appropriate output components identified in all instances where required	Appropriate output components identified where required – minor shortcomings	Appropriate in most instances where required. Choices not always substantiated.	Not done/not appropriate output components identified	**3**	
General	**Addresses the requirements specified in phase 1**					
	2		**1**	**0**	**2**	
	All requirements addressed		At least 50%	Less than 50%		
				Total:	**30**	

Assessment for phase 3: **Name of learner:**_____

NB: Phase 3 and 4 will only be assessed once phases 1 and 2 have been completed AND phases 1 and 2 are related to the topic covered in phases 3 and 4

	Coding and Implementation: Criteria					Possible Mark	Mark Obtained
Database	**Database: tables**					**4**	
	4	**3**	**2**	**1**	**0**		
	Number of tables speaks to an effective solution – tables normalised (Appropriate number of tables to support effective solution) Primary keys and foreign key in related tables effective and appropriate. All the relationships well defined using the correct primary and foreign keys	Number of tables speaks to a good solution At least one effective and appropriate relationship using the correct primary and foreign keys. Some of the primary keys or foreign key in related tables not appropriate	Number of tables indicates partial solution – tables not normalised (Could have more tables for a more effective solution) Most of the primary keys or foreign keys not appropriate or only default keys used.	Number of tables irrelevant to solution. (Only one table/ a number of irrelevant tables with repetition of fields in tables. Or more than one database with one table in each database instead of one database)	No database used		
	Field types					**2**	
	2		**1**		**0**		
	All fields with appropriate data types		Data types of some of the fields not appropriate		Only default types used or no database used		
	Field sizes - database					**2**	
	2		**1**		**0**		
	All fields with appropriate field sizes		Some of the fields are too large/small		Only default field sizes used or no database used		
	Descriptive field names in database					**1**	
Programming solutions	**NOTE: Assess the code if the program does not run. This section has to do with the programming in Delphi/Java <u>excluding SQL</u>**						
	User defined data structures (<u>excluding database tables</u>)					**3**	
	3	**2**		**1**	**0**		
	Used appropriate and most effective data structures to solve the problem in all instances	Used appropriate and most effective data structures in most instances or with minor shortcomings		Appropriate and most effective use of data structures in less than 50% of the instances	Inappropriate or ineffective use of data structures		
	Variables/data structure names meaningful					**2**	
	2		**1**		**0**		
	Variable/data structure names meaningful in all instances, throughout program		Minor shortcomings		Not meaningful		
	Selection structures					**3**	
	3	**2**		**1**	**0**		
	Used appropriate and most effective selection structures to solve the problem in all instances	Used appropriate and most effective selection structures in most instances		Inappropriate or ineffective use of selection structures in most instances	Not used		

	Repetition structures					
	3	**2**	**1**	**0**		
	Used appropriate and most effective repetition structures to solve the problem in all instances	Used appropriate and most effective repetition structures in most instances	Inappropriate or ineffective use of repetition structures in most instances	Not used	**3**	
	Complex programming techniques (e.g. play video clips, borrowed code, threads, networking)					
	3	**2**	**1**	**0**		
	It works correctly. Appropriately used and adds value to the solution	It works correctly. Not always appropriately used or does not really add value to the solution	Effort made but it does not work properly. Inappropriately used. Not relevant to the solution	No complex techniques used or exceeds more that 10% of code	**3**	
	Data flow and processes (user defined parameter passing)					
	3	**2**	**1**	**0**		
	Excellent interaction/ communication between modules/classes. Includes advanced data types as parameters or return types	Proficient/adequate /some communication between modules/classes with small flaws. Includes some parameter passing between modules.	Limited communication between modules/classes. Only primitive data types passed as parameters	No communication between modules/classes. No parameters passed	**3**	
	Re-use of code (Classes and methods/subprograms)					
	3	**2**	**1**	**0**		
	Appropriate and effective re-use of code and/or methods. Modules independent. Local variables used where applicable. Global variables only used when required.	Re-use of code and/or methods but not always appropriate/effective Modules not always independent. Some global variables restrict independence of modules.	Re-use of code inappropriate/not effective. Modules could have been broken down into more modules. Almost all the variables declared globally.	Linear programming – one continuous program, no modules. No re-use of code and/or methods.	**3**	
	Solution algorithms					
	3	**2**	**1**	**0**		
	All solution algorithms used in solving problem are appropriate and effective.	Appropriate and effective with minor shortcomings	Mostly inappropriate or not effective	Totally inappropriate solution algorithms or mostly ineffective	**3**	
	Correctness of solution algorithms					
	3	**2**	**1**	**0**		
	No logical error. All the results of processing are correct.	Minor shortcomings Very few minor logical errors. Very few of the results are not correct.	Logical errors Many results incorrect	Many logical errors. Almost all the results are incorrect/few of the required results are delivered	**3**	
Input	**Input strategies**					
	3	**2**	**1**	**0**		
	Most appropriate, effective input strategies used in all instances, e.g. input from text files, database tables, user input	Appropriate and effective with minor shortcomings	Some strategies could have been more appropriate/effective	Mostly inappropriate or not effective	**3**	
Output	**Output vs. requirements**					
	3	**2**	**1**	**0**		
	Output meets all the requirements for the solution	Output meets most of the requirements for the solution.	Output meets less than 50% of the requirements of the solution	No output	**3**	

	Structures for output (coding)				**3**	
	3	**2**	**1**			
	Output always well-structured, readable with headings and subheadings. Headings repeated on following page/screen where applicable.	Output in most of the cases well structured, readable with headings and subheadings. Headings repeated on following page/screen in most of the cases where applicable.	Output not well structured. Headings and or subheadings in most of the cases not well formulated or absent. Headings mostly not repeated on following page where applicable.	No code to display output		
	Format of output – all values formatted appropriately where applicable, e.g. currency, units of measurement, etc.				**1**	
Database interaction	**Database connection - string/path set and work correctly**				**1**	
	Database interaction with program				**2**	
	2	**1**	**0**			
	Database interacts with program in a meaningful way e.g. queries and reports	Not always meaningful	No interaction / not meaningful			
	Manipulate records correctly via SQL (2 marks for each correct SQL-statement to a maximum of 6 marks)				**6**	
	Insert, delete, select, update. List other:					
	Manipulate fields via SQL (2 marks for each correct SQL-statement to a maximum of 6 marks)				**6**	
	Calculations on fields, change contents, show only one field, named fields, sort according to fields. List other:					
Error handling and Testing	**Program compiles successfully – no syntax errors**				**1**	
	Output errors				**3**	
	3	**2**	**1**	**0**		
	No run time errors. All the options are executed successfully	Some of the options produce errors when executed	Only one or two of the options can be executed successfully	None of the options execute successfully		
	Input validation				**3**	
	3	**2**	**1**	**0**		
	All input that should be validated is validated using code	Most of input that should be validated is validated	Less than 50% of input that should be validated is validated.	No validation		
	Error messages				**3**	
	3	**2**	**1**	**0**		
	Appropriate and user friendly error messages in all cases where data validation is applied	Appropriate and user friendly error messages in most of the cases where data validation is applied	Appropriate and user friendly error messages in very few of the cases where data validation is applied	No error messages		
	Program gives outpur (output is the result of processing, i.e. GUI windows with no functionality do not classify as output)				**2**	
	Correctness of output				**3**	
	3	**2**	**1**	**0**		
	Program gives correct as well as appropriate output in all cases.	Program gives correct and appropriate output in most cases.	Program gives correct and appropriate output in less than 50% of the cases. .	No output or only incorrect output		
	Exception handling				**2**	
	2	**1**	**0**			
	Used try… catch exception handling	Attempted exception handling or used if statements to handle error cases	No attempt			
User Interface	**Different screens used (windows/panels/tab sheets, etc) appropriately**				**1**	
	Components used for input/output (implemented as per phase 2 OR appropriate changes made from Phase 2 feedback)				**2**	
	2	**1**	**0**			

	Always appropriate, most effective	Minor shortcomings	Mostly not appropriate / effective		
	Labels/prompting with exact formats for input				
	2	**1**	**0**	**2**	
	Applied constantly throughout the entire project where required	Applied in most of the cases where required	Mostly not applied		
	Consistent layout (same look and feel throughout)				
	2	**1**	**0**	**2**	
	For all screens (same colours, fonts used throughout program)	Most of the screens (some screens different colours, fonts etc) – minor shortcomings	Layout mostly inconsistent		
	Informative output/reports are informative and easy to read and interpret (				
	2	**1**	**0**	**2**	
	All output, all screens are informative and easy to read and interpret (appropriate font size, layout, colour, etc.)	Most of the output are informative and easy to read and interpret – minor shortcomings	Output mostly not informative or not easy to read and interpret		
	Grouping of input / output				
	2	**1**	**0**	**2**	
	Type of input/output grouped together, e.g. address information, for all screens	Type of input/output mostly grouped together – minor shortcomings	Not grouped together in most instances		
	Navigation between screens				
	2	**1**	**0**	**2**	
	Easy to navigate between screens – logical flow of events	Easy and logical with minor shortcomings	Not easy or no logical flow		
	Help available as part of the program				
	2	**1**	**0**	**2**	
	Help available as part of the program, works correctly with clear instructions	Not always available, clear or does not always work correctly	Not available		
	Context sensitive				
	2	**1**	**0**	**2**	
	Context sensitive help available (tool tip text included), appropriate and effective	Not always appropriate or effective	Not available		
	Design vs. target user				
	2	**1**	**0**	**2**	
	Design considers target user (age, literacy level, visual impairments, appropriateness of images, etc) Design appropriate for target user	Consideration given, but minor shortcomings	Totally inappropriate		
General	**Internal documentation**				
	2	**1**	**0**	**2**	
	Code is commented/annotated to explain/describe for easy interpretation throughout the program	Commented/annotated but not throughout the program	Not commented/annotated		
	Separation of sections				
	2	**1**	**0**	**2**	
	Sections in the code of the program clearly separated to enhance readability (spacing, comments for method/subsection, etc)	Some sections separated	Not separated		
				Total:	**100**

Assessment for phase 4: **Name of learner:**_____

	Documentation and Evaluation: Criteria				Possible Mark	Mark Obtained
Technical documentation	**Database design clearly shown and explained**				1	
	Description of other data structures used				1	
	Describe/show relationship between modules/programs				1	
	Description of unusual coding – functions, calculations, borrowed code, etc.				1	
	Specification of test procedures and test results				1	
	Error recovery and troubleshooting clearly described				1	
	Examples of sample runs with results				1	
	Source code available				1	
	List of data files that the program consists of is provided				1	
User Guide	**Title sheet**				1	
	Table of contents				1	
	Background to the project (personal motivation and choice)				1	
	Introduction to the project (genre)				1	
	Project scope and limitations are clearly described				1	
	User input requirements/formats				1	
	Output/reports produced (screen dumps included)				1	
	Detailed instructions and walk-through				3	
	3	**2**	**1**	**0**		
	All the steps to be followed when using the program clearly indicated and supported by screen dumps where required. Well structured with a logical flow. Well formulated – easy to understand.	Most of the steps to be followed when using the program clearly indicated and some screen dumps where required. Not always well structured and not always a logical flow. Not always well formulated	Only a few of the steps to be followed when using the program is indicated. No screen dumps. Not well structured. Not well formulated.	Not done		
	Hardware required to run program (1 mark each up to max of 3 marks)				3	
	Hard drive space, memory, CPU, special devices – Give list **List:**					
	Software requirements listed				1	
	Installation instructions given				1	
	List of required files				1	
	User troubleshooting clearly described				1	
	Future developments/extensions described				1	
	Acknowledgement where required				1	
General Evaluation	**Time Management**				3	
	3	**2**	**1**	**0**		
	All deadlines met – all 4 phases and all the required work were done.	Met 3 deadlines or submitted on time but some of the work was not done.	Met only two deadlines or submitted on time but most of the work was not done.	Always late, never done		
	Ability to explain purpose of working of randomly selected code				3	
	3	**2**	**1**	**0**		
	Explained all selected code clearly and with confidence Shows excellent insight	Explained selected code with minor shortcomings Shows insight	Not able to explain most of the selected code appropriately. Lacks insight	Unable to explain		

	Real-life application of system					
	3	**2**	**1**	**0**	**3**	
	The solution is a working system that can be applied in a real situation	The solution is a system that can be applied in a real situation with some fine tuning	Some parts can be applied in the real situation	Totally irrelevant Will not work in real situation		
				Total:	**37**	