2008 HSC Notes from the Marking Centre
Software Design and Development

Published by Board of Studies NSW
GPO Box 5300
Sydney 2001
Australia

Tel: (02) 9367 8111
Fax: (02) 9367 8484
Internet: http://www.boardofstudies.nsw.edu.au

2009083

Contents

2008 HSC NOTES FROM THE MARKING CENTRE SOFTWARE DESIGN AND DEVELOPMENT

Introduction

This document has been produced for the teachers and candidates of the Stage 6 course in Software Design and Development. It contains comments on candidate responses to the 2008 Higher School Certificate examination, indicating the quality of the responses and highlighting their relative strengths and weaknesses.

This document should be read along with the relevant syllabus, the 2008 Higher School Certificate examination, the marking guidelines and other support documents which have been developed by the Board of Studies to assist in the teaching and learning of Software Design and Development.

Teachers and candidates should be aware that examiners may ask questions in Sections I and II that combine knowledge, skills and understandings from across the core of the HSC syllabus.

Section I

Question	Correct response
1	A
2	B
3	D
4	A
5	B
6	C
7	D
8	A
9	C
10	C

Question	Correct response
11	D
12	C
13	A
14	C
15	D
16	B
17	B
18	A
19	C
20	B

Section II

General comments

The 2008 Higher School Certificate examination in Software Design and Development required candidates to analyse and interpret situations and to apply their knowledge to these situations. Many candidates showed a sound understanding of concepts but were less able to apply this knowledge appropriately, often giving general answers or answers not directly related to the particular situation described in the question. It should be noted by candidates that if a scenario is given in the question, then it should be referred to in their responses. Candidates should relate their knowledge of the concept being examined to the situation or system described in the question.

Question 21

(a) (i) Better responses correctly identified a feature of reverse engineering. Weaker responses tended to identify superficial features only or provided single word answers such as 'decompilation'.

(ii) Better responses explained the effect of reverse engineering in relation to copyright, financial and legal issues. Weaker responses focused on the process rather than its specific effect on intellectual property rights.

(b) (i) Better responses included appropriate screen elements within the design and correctly labelled the diagram illustrating an understanding of screen design principles. Candidates are reminded that screen designs should be drawn clearly and large enough to identify both the fields and particular interface design elements included.

Weaker responses provided a simplistic layout, with minimal use of screen elements such as radio buttons, check boxes or combo-boxes, and often did not provide labelling or annotation of the diagram.

(ii) Better responses included the correct use of columns for data type, field size, description and example.

Weaker responses simply listed the identifiers and used incorrect or inappropriate data types such as 'menu' or 'drop down' instead of using terms such as 'numeric', 'string' or 'Boolean'.

(c) (i) Mid-range responses identified the variable as a counter. Poorer responses showed little understanding of the algorithm or the use of the variable 'fishfingers'. Candidates are reminded that they should be able to read and interpret algorithms expressed either as flowcharts or in pseudocode.

(ii) Weaker responses incorrectly identified 'number' or 'fishfingers' as the sentinel value, instead of the correct answer of −1.

(iii) Better responses outlined the need to test all possible paths through the algorithm. Weaker responses did not mention 'boundary conditions', 'different data types' or 'invalid data'. Better responses not only included the above examples, but correctly linked these to the

scenario. Candidates are reminded that questions such as these require specific answers relating directly to the given algorithm rather than broad general features.

(iv) Better responses demonstrated a clear understanding that the desk check would terminate when the sentinel value '−1' was input, hence making the rest of the test data set redundant. They also showed the values of all variables as they changed, using an appropriate layout for a desk check.

Weaker responses attempted to use all of the test data, and some had little understanding of the general layout, purpose and process of a desk check. They simply placed all given data items in the number column reflecting a limited understanding of either the algorithm or the process of desk checking. Some responses showed obvious understanding of the algorithm, and provided the final correct outputs in the table but did not demonstrate the steps taken to reach that output as required by the question 'completing a desk check'.

(v) Better responses mentioned the need to replace the post-test loop with a pre-test loop, modified the condition detecting the 'small' value and identified the irrelevance of the final 'IF' statement.

Weaker responses either copied sections of the algorithm into their writing booklets, or identified specific line numbers only and did not attempt to explain what the cause of the error was in the specified line.

Question 22

(a) (i) Better responses identified the fact that accountNumber was used as the linear search variable to find the name and address of the person in the relevant account record.

Weaker responses only used terms such as 'match' or 'locate' without any description of what was being matched or located. The weakest responses did not attempt to relate the use of the variable in the given algorithm and used generalisations such as 'stores a value'.

(ii) Better responses identified the error and then went on to explain that the omission of an 'IF' condition caused an infinite loop. Mid-range responses correctly identified that the error was in line 150. However, their description of the error was inaccurate. They either thought that 'WHILE NOT' is an invalid pseudocode keyword combination, or that 'NOT found' should have been written as 'Found = false'. Candidates are reminded that 'WHILE NOT found' is a perfectly acceptable use of a Boolean variable such as 'found'.

Weaker responses incorrectly commented on the fact that the word 'THEN' in line 170 was in the wrong line and should have been at the end of line 160. Other weaker responses showed concern that there was no new data read inside the loop – failing to recognise the purpose of line 210.

(iii) Better responses identified the data found in the algorithm and included appropriate print statements between either lines 190 and 200, or lines 230 and 240. Some weaker responses incorrectly enclosed their variable names inside quotation marks in the print statement.

(b) (i) Better responses included the use of terminology such as 'documentation' or 'guidelines', correctly indicating the purpose of design specifications. They then included an explanation of what was included in these specifications, with some providing examples.

Weaker responses consisted of very short broad answers that did not provide demonstration of sufficient understanding. Other weaker responses confused the feasibility study with design specifications.

(ii) Better responses referred to the usage of the design specifications at the testing and evaluating stage. They demonstrated an understanding that this documentation provided guidelines that would be used during the quality assurance assessment process with the requirement that all guidelines must be met if the finished product is to meet the set benchmark. Some better responses included the fact that for a system to reach the established benchmark during the quality assurance process, the design specifications must be continually referred to and adhered to throughout all the subsequent stages of the program development cycle.

Weaker responses indicated some knowledge about quality assurance without relating it to the design specifications or merely attempted a description of design specifications without referring to quality assurance.

(iii) Better responses included relevant reasons for the communication.

Weaker responses referred to communication with users at various stages in the development process rather than specifically in the development and modification of design specifications.

(iv) Better responses identified and described the benefits of using CASE tools at this stage. They included specific features of CASE tools and ways in which these features could be productively used rather than just using general terms such as 'quicker' or 'easier'.

Weaker responses only indicated some facts about CASE tools but did not relate this knowledge to the question. Weaker responses also inappropriately referred to the use of CASE tools in the implementation stage rather than in the development of design specifications.

(c) (i) Better responses included answers to both parts of this question. They identified a relevant issue such as problems that could occur if the system was only available in English. They then went on to discuss what must be included in the design and eventual software solution for the problem to be effectively addressed.

Weaker responses identified a social or ethical issue with no reference to the implications for the design of the software.

(ii) Many mid-range responses indicated knowledge of the responsibilities of a software developer, but did not attempt to relate these responsibilities to the described proposed Tax Office system.

Better responses included a discussion of a variety of responsibilities rather than just stating one or two. They also went beyond the solving of social and ethical issues and included responsibilities such as adhering to a code of conduct or provision of documentation for maintainability if the tax requirements are changed.

Question 23

(a) (i) Better responses included the actual formulae for the calculations in the processes. Candidates are reminded that the correct format for an IPO chart can be found in the 'Software and Course Specifications' document.

(ii) Better responses indicated correctly that the stub was the Calc_Circ function, as it merely displays the text 'Circle' without yet including the logic to perform the detailed calculations required.

(iii) Better responses took the required statements from the Calc_Rectangle function provided in the question to generate the appropriate pseudocode. Weaker responses did not use the symbol * to indicate multiplication and simply rewrote the formulae provided in the question. Other weaker responses included extra unnecessary variables in the input statement in addition to radius, or did not construct the return message correctly.

(b) (i) In the better responses, candidates provided both the line number(s) for the nominated control structure and named it correctly. However, in many mid-range responses candidates could only locate a control structure but not name it correctly, or gave only a partial name such as 'loop' instead of 'pre-test repetition'. There was confusion in some responses between control structures and data structures.

Candidates are advised that line numbers are used in algorithms to allow them to easily refer to specific statements, and the line numbers should be used specifically for this purpose.

(ii) Better responses correctly identified this as a linear search. Weaker responses confused searches with sorts or did not understand the difference between linear and binary searches. Some responses identified the search technique as 'sequential', however this is not a term used in the syllabus.

(iii) Better responses introduced a flag by initialising a Boolean variable to false between 110 and 120, and then changing the flag to true inside the 'IF' statement (140 to 160). After the repetition (ie after line 190) the flag is tested, with an appropriate output statement if the flag is true.

Weaker responses modified the IF-ENDIF selection structure by incorrectly adding an ELSE section and displaying a message at that point. Unfortunately, this will print a message for every dailyRainfall value that does not match the inputted targetRainfall value.

Some responses included a rewrite of the entire algorithm from the question. This was not necessary. Examples of acceptable methods of adding lines of code include statements such as: 'Add line 115: Found = False' or 'Add after line 110: Found = False'.

(iv) Better responses demonstrated an understanding of what a standard module is and why software developers include them in their developed code. The better responses clearly related the features of standard modules to the corresponding benefits for developers.

Weaker responses did not distinguish between the advantages of a modular approach to programming and the advantages of using standard modules.

(c) (i) In better responses, candidates recognised the statement given in the question as a 'User Input Statement', and were then able to refer to the required syntax definitions of a 'Variable List', 'Variable' and 'Letter' to show that the statement was legal. Weaker responses simply mentioned the need to refer to the definition of a 'Variable List' without reference to other definitions, or did not refer to the syntax and statement definitions at all.

(ii) Better responses realised that the use of the terms IF and ENDIF were incorrect, based on the definition of the Conditional Statement provided. Better responses described the other two syntax errors involving the invalid use of ':=' and 'Tom' by referring to the EBNF definitions for Condition and Constant.

(iii) While mid-range responses showed an understanding of the use of railroad diagrams, only better responses were able to demonstrate sufficient understanding by constructing an appropriate diagram for a post-test repetition structure using the syntax defined in the question. Better responses also used terminal symbols for the reserved words REPEAT and UNTIL and included a 'Conditional Exp' after the UNTIL. They also included a choice and repetition of statements between REPEAT and UNTIL.

Section III

Question 24 – Evolution of programming languages

(a) (i) Better responses showed an understanding that paradigms are a way of thinking about or approaching problem solving.

(ii) Better responses identified a language that supports object oriented programming (OOP). Some responses stated, incorrectly, that languages that offer visual drag and drop facilities such as Visual Basic are object-oriented languages.

(iii) Better responses identified similarities between the paradigms. Mid-range responses demonstrated some knowledge of the two paradigms and in the process were able to demonstrate differences. Weaker responses presented answers in broad terms using technical language referring to the building blocks of each paradigm, but with little specific detail. Often responses omitted to give examples of the type of solutions for which the paradigm would be suitable.

(b) (i) Better responses showed understanding that the inference engine provides the logic to work through the facts and rules to arrive at a solution or goal.

(ii) Better responses succinctly listed all of the information. They correctly interpreted the facts and rules, and made the required connections to obtain a simple final conclusion.

Weaker responses could only identify the facts and could not apply the rules, with these responses simply rewriting the rules given in the question, and filling in or attempting to fill in the variables in the rules with names.

(c) (i) Weaker responses had difficulty distinguishing between LFIND and RFIND, and did not show the working required to get to their final answer.

(ii) The same comments for part (c) (i) also apply to this part, with the provision of the required working even more critical in deriving the correct answer.

(d) (i) In better responses, candidates demonstrated an understanding of the concept of instances by providing a number of relevant examples of instances. They provided sample data to show that the attributes would change for each instance, such as a client user who is different to an administrator user. Weaker responses confused an instance with the methods listed in the sample pseudocode.

(ii) Mid-range responses described the correct sequence of processes in the sample pseudocode, but without relating these correctly to an object-oriented environment. In better responses, candidates described internal and external message passing between the objects. In weaker responses, many candidates confused external message passing with screen prompts to the user.

(iii) In mid-range responses candidates demonstrated appropriate knowledge of the concept of inheritance. In better responses, candidates applied their understanding of inheritance to the scenario provided, supplementing their answer with a relevant diagram to show that the properties of a teacher are inherited from those of a human being, but that a teacher also can have specific properties not owned by a human being (such as possessing a teaching degree for example).

Question 25 – The software developer's view of the hardware

(a) (i) Better responses correctly divided the two binary numbers, showing the appropriate working. Mid-range responses derived the correct answer by converting to a decimal, evaluating and converting the answer back to binary. Candidates are reminded to read the question carefully as some added rather than divided the two numbers given.

(ii) In the better responses, candidates provided good descriptions of both floating point and fixed point methods. Mid-range responses outlined only one method of representing fractions in binary and often without sufficient description to fully describe the process. Candidates are reminded that the use of an example often assists in demonstrating understanding and the better responses included examples of both methods. A small number of weaker responses referred incorrectly to the use of one and two's complement representation.

(iii) In better responses that demonstrated a good grasp of memory addressing, candidates were able to recognise the need to store each 8 bit ASCII character in a separate memory location and correctly incremented the address of each location through 2F to 30 and 31. In weaker responses, candidates reduced each ASCII letter to two characters either

arbitrarily in binary or by converting to hexadecimal and leaving the resultant 8 character string in memory location 2E.

(b) (i) Better responses provided a clear understanding of the purpose of a truth table as displaying the inputs and resultant outputs for each logic gate in a circuit diagram.

(ii) Better responses demonstrated a good understanding of the operation of the given circuit by drawing a table with correctly labelled columns and including all possible input values for *A* and *B*. Weaker responses appeared to give random incorrect outputs for a given gate or provided a set of outputs for gates not included in the circuit such as XOR or AND gates.

(c) (i) In better responses, candidates recognised that *R* and *S* were data streams passing through an AND gate to produce a single output data stream. Candidates are reminded to show the processes and working involved; in this case a bit-by-bit comparison was required.

(ii) Mid-range responses successfully deduced at least one correct value of *S* to produce the stated output. A significant number of better responses stated all four possible values required to answer the question in full.

(d) (i) Better responses, showing a good understanding of the reason for including a header in the data packet, listed the possible contents, with mid-range responses listing at least two components correctly. Some weaker responses confused word processing headers with headers in a data packet.

(ii) Better responses showed an understanding of the format and contents of the data packets transmitted from each device, and were able to compare the component parts of the data stream effectively, including header, data and trailer. Comparisons used in these responses gave an indication of similarities such as the identification of the device in the header together with some indication of the start of the data stream. Good responses also included differences such as the nature of the data transmitted and the relative size of the packets transmitted and the need for different error checking methods in the trailer. Weaker responses tended to provide descriptions of how the devices were used rather than describing the contents of the data packets being sent. Some weaker responses confused the term USB with a USB memory stick and so provided an incorrect response.

Software Design and Development

2008 HSC Examination Mapping Grid

Question	Marks	Content	Syllabus outcomes
Section I			
1	1	9.1.1	H3.1
2	1	9.3	H5.1, H5.2
3	1	9.2.1	H5.2
4	1	9.2.1	H5.2
5	1	9.2.1	H5.2
6	1	9.1.2	H5.1
7	1	9.3	H5.2, H6.2
8	1	9.2.3	H1.3
9	1	9.2.3	H4.2
10	1	9.2.2	H5.2
11	1	9.2.3	H4.2
12	1	9.2.3	H1.3
13	1	9.2.4	H5.2, H5.3
14	1	9.2.2	H1.3
15	1	9.2.3	H1.1, H1.3
16	1	9.1.2	H4.2
17	1	9.1.1	H3.1
18	1	9.1.1	H2.2
19	1	9.2.3	H4.2, H4.3
20	1	9.2.2	H4.2
Section II			
21(a)(i)	1	9.1.1, 9.2.1	H3.1, H3.2
21(a)(ii)	2	9.1.1, 9.2.1	H3.1, H3.2
21(b)(i)	4	9.2.1	H4.1, H4.2
21(b)(ii)	3	9.1.2	H4.2, H5.2
21(c)(i)	1	9.2.1, 9.2.2, 9.2.3	H4.2, H4.3
21(c)(ii)	1	9.2.1, 9.2.2, 9.2.3, 9.3	H4.2, H4.3
21(c)(iii)	2	9.2.4	H4.1, H4.2
21(c)(iv)	3	9.2.1, 9.2.2, 9.2.3, 9.2.4, 9.3	H4.1, H4.2
21(c)(v)	3	9.2.1, 9.2.2, 9.2.3, 9.3	H4.1, H4.2
22(a)(i)	2	9.2.2	H4.2, H4.3
22(a)(ii)	2	9.2.2	H4.2, H4.3
22(a)(iii)	2	9.2.2	H4.2, H4.3
22(b)(i)	2	9.2.1	H4.2, H4.3
22(b)(ii)	2	9.2.1	H4.2, H5.2
22(b)(iii)	2	9.2.1	H4.1, H4.2, H6.1

Question	Marks	Content	Syllabus outcomes
22(b)(iv)	2	9.2.1, 9.2.2	H4.1, H4.2
22(c)(i)	3	9.1.1	H3.1
22(c)(ii)	3	9.1.2	H1.2, H4.1, H5.1
23(a)(i)	3	9.2.1	H5.2
23(a)(ii)	1	9.2.3	H4.2, H4.3
23(a)(iii)	3	9.2.2	H5.2
23(b)(i)	1	9.2.2	H4.2, H4.3
23(b)(ii)	1	9.2.2	H4.2, H4.3
23(b)(iii)	2	9.2.2	H4.2, H4.3
23(b)(iv)	3	9.2.2	H4.2, H4.3
23(c)(i)	1	9.2.3	H4.2, H4.3
23(c)(ii)	2	9.2.3	H4.2, H4.3
23(c)(iii)	3	9.2.3	H4.2, H4.3
Section III			
24(a)(i)	1	9.4.1	H1.2, H2.1
24(a)(ii)	1	9.4.1	H1.2, H2.1
24(a)(iii)	3	9.4.1	H4.2
24(b)(i)	1	9.4.1	H2.1, H2.2
24(b)(ii)	3	9.4.1	H4.2
24(c)(i)	2	9.4.1	H4.2
24(c)(ii)	2	9.4.1	H4.2
24(d)(i)	2	9.4.1	H1.2
24(d)(ii)	3	9.4.1	H1.2, H4.1
24(d)(iii)	2	9.4.1	H1.2, H4.2
25(a)(i)	2	9.4.2	H1.1, H1.3
25(a)(ii)	3	9.4.2	H1.1, H1.3
25(a)(iii)	2	9.4.2	H1.1
25(b)(i)	1	9.4.2	H4.1
25(b)(ii)	3	9.4.2	H1.1, H4.1
25(c)(i)	2	9.4.2	H1.1, H1.3
25(c)(ii)	2	9.4.2	H1.1, H1.3
25(d)(i)	2	9.4.2	H1.1, H1.3, H4.1
25(d)(ii)	3	9.4.2	H1.1, H1.3, H4.1

BOARD OF STUDIES
NEW SOUTH WALES

2008 HSC Software Design and Development Marking Guidelines

The following marking guidelines were developed by the examination committee for the 2008 HSC examination in Software Design and Development, and were used at the marking centre in marking student responses. For each question the marking guidelines are contained in a table showing the criteria associated with each mark or mark range. For some questions, 'Sample Answers' or 'Answers may include' sections are included. These are developed by the examination committee for two purposes. The committee does this:

(1) as part of the development of the examination paper to ensure the questions will effectively assess students' knowledge and skills, and

(2) in order to provide some advice to the Supervisor of Marking about the nature and scope of the responses expected of students.

The examination committee develops the marking guidelines concurrently with the examination paper. The 'Sample Answers' or similar advice are not intended to be exemplary or even complete answers or responses. As they are part of the examination committee's 'working document', they may contain typographical errors, omissions, or only some of the possible correct answers.

The information in the marking guidelines is further supplemented as required by the Supervisor of Marking and the senior markers at the marking centre.

A range of different organisations produce booklets of sample answers for HSC examinations, and other notes for students and teachers. The Board of Studies does not attest to the correctness or suitability of the answers, sample responses or explanations provided. Nevertheless, many students and teachers have found such publications to be useful in their preparation for the HSC examinations.

A copy of the Mapping Grid, which maps each question in the examination to course outcomes and content as detailed in the syllabus, is also included.

Section II

Question 21 (a) (i)

Outcomes assessed: H3.1, H3.2

MARKING GUIDELINES

Criteria	Marks
• Identifies a feature of reverse/backwards engineering	1

Sample answer/Answers could include:

A feature of reverse/backward engineering is the legality of the process. If the code has copyright then the reverse/backward engineering is illegal.

Question 21 (a) (ii)

Outcomes assessed: H3.1, H3.2

MARKING GUIDELINES

Criteria	Marks
• Explains how reverse/backwards engineering affects the intellectual property rights of software developers	2
• Identifies features of intellectual property rights of software developers	1

Sample answer/Answers could include:

Intellectual property rights apply to the ideas of the software developer.

If people use reverse/backwards engineering to steal ideas from software developers the developers will be less likely to develop future projects and share their ideas.

Unless the developer releases their intellectual rights it is illegal to use reverse/backwards engineering.

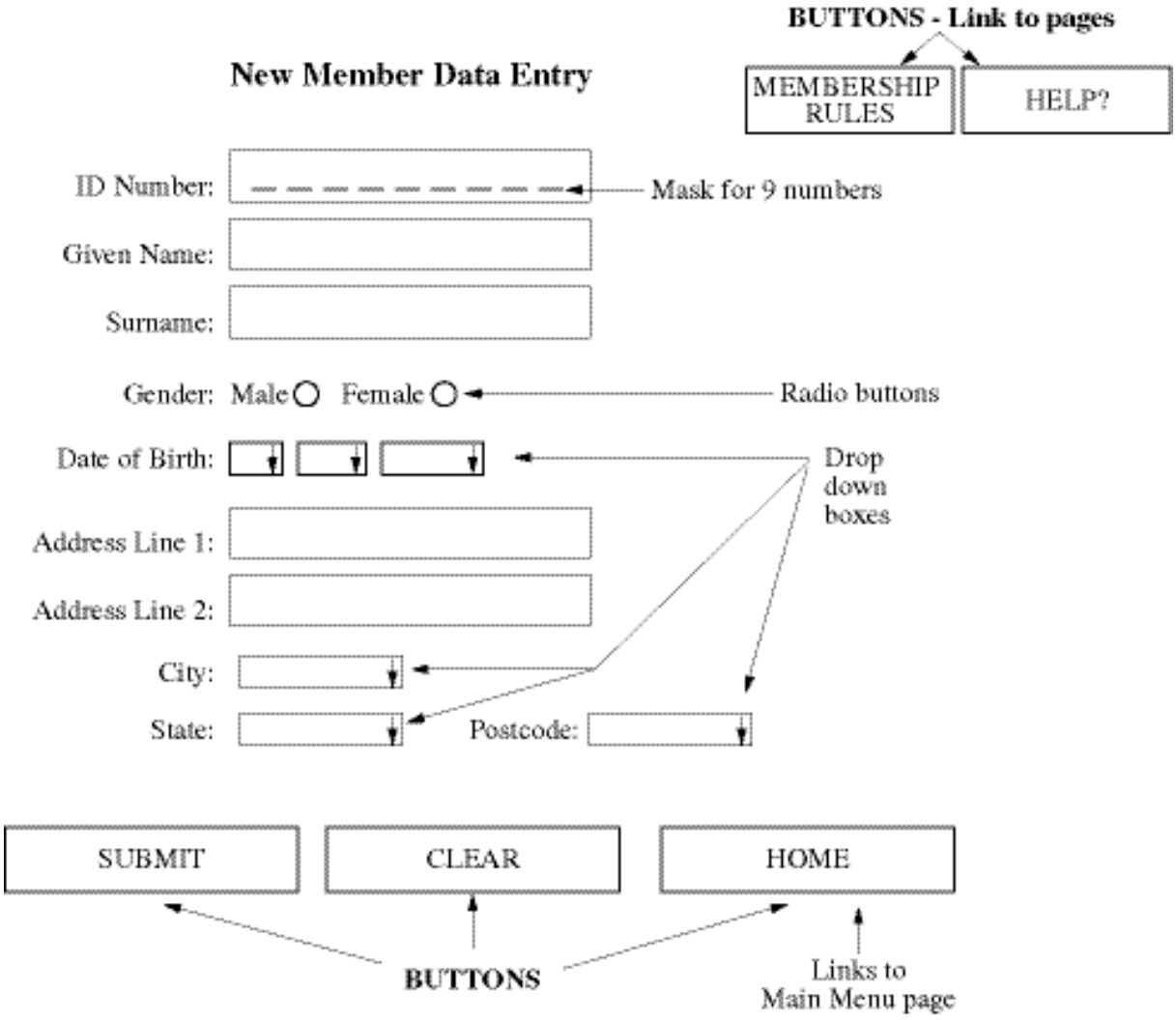
Question 21 (b) (i)

Outcomes assessed: H4.1, H4.2

MARKING GUIDELINES

Criteria	Marks
• Provides an appropriate design which involves most labels of key design consideration	4
• Provides a design including some labels of key design considerations	3
• Provides a design	2
• Attempts a design	1

Sample answer/Answers could include:



Question 21 (b) (ii)

Outcomes assessed: H4.2, H5.2

MARKING GUIDELINES

Criteria	Marks
• Provides a substantially correct data dictionary	3
• Provides a data dictionary relating to the question	2
• Identifies features of a data dictionary	1

Sample answer/Answers could include:

Field Name	Field size	Data Type	Description
ID-number	9	String	Primary key
Surname	30	String	Family name eg Smith
Given Names	30	String	Name eg Sue, Tom
Gender	1	Boolean	Either M or F
Date of Birth	10	Date	eg 26/10/2000
Address line 1	30	String	eg Villa 5
Address line 2	30	String	eg 11 Smith St
City	30	String	eg Chatswood
State	30	String	eg Queensland
Post Code	4	Numeric	eg 2141

Question 21 (c) (i)

Outcomes assessed: H4.2, H4.3

MARKING GUIDELINES

Criteria	Marks
• Identifies the purpose of the variable 'fishfingers'	1

Sample answer/Answers could include:

The purpose of 'fishfingers' is to count the READ (entered) numbers.

Question 21 (c) (ii)

Outcomes assessed: H4.2, H4.3

MARKING GUIDELINES

Criteria	Marks
• Identifies the sentinel value in the algorithm as –1	1

Sample answer/Answers could include:

Answer is –1.

Question 21 (c) (iii)

Outcomes assessed: H4.1, H4.2

MARKING GUIDELINES

Criteria	Marks
• Describes aspects to be considered when designing test data for the algorithm	2
• Identifies a feature of test data	1

Sample answer/Answers could include:

Positive and negative integers, rational numbers, 0 and the sentinel value –1. Include equal values and characters other than numbers should also be entered.

Question 21 (c) (iv)

Outcomes assessed: H4.1, H4.2

MARKING GUIDELINES

Criteria	Marks
• Provides substantially correct desk check	3
• Provides desk check relating to the question	2
• Identifies features of a desk check	1

Sample answer/Answers could include:

fishfingers	Number	Large	Small	OUTPUT
0	7	7	7	Enter a Mark or –1
1	5		5	Enter a Mark or –1
2	5			Enter a Mark or –1
3	–1		–1	7, –1, 3

Question 21 (c) (v)

Outcomes assessed: H4.1, H4.2

MARKING GUIDELINES

Criteria	Marks
• Identifies errors and provides a correct solution to the errors	3
• Identifies error and provides an attempt at a solution	2
• Identifies errors OR • Provides an attempt at a solution	1

Sample answer/Answers could include:

* Pre-test loop needed
160 While number < > –1
* Marks need to be entered
270 IF fishfingers > 0 THEN
*Small always –1
– change to pre-test loop
– move 170 to after 250

Question 22 (a) (i)

Outcomes assessed: H4.2, H4.3

MARKING GUIDELINES

Criteria	Marks
• Provides explanation as to how accountNumber has been used	2
• Identifies features of the variable accountNumber	1

Sample answer/Answers could include:

The variable accountNumber has been used as a search variable to find the person's name and address who owns the account.

The variable is used to match with the person's account in the search process.

Question 22 (a) (ii)

Outcomes assessed: H4.2, H4.3

MARKING GUIDELINES

Criteria	Marks
• Provides description of error that indicates understanding of the problem	2
• Identifies error	1

Sample answer/Answers could include:

The error is: If not found, the algorithm loops indefinitely.

Question 22 (a) (iii)

Outcomes assessed: H4.2, H4.3

MARKING GUIDELINES

Criteria	Marks
• Provides substantially correct alteration to algorithm that prints data on the screen	2
• Attempts alteration to algorithm indicating limited understanding	1

Sample answer/Answers could include:

Between lines 190 and 200, add

```
PRINT     'Account number is'    account Number,
          'Account Name is'      account Name,
          'Account Address is'   account Address.
```

Question 22 (b) (i)

Outcomes assessed: H4.2, H4.3

MARKING GUIDELINES

Criteria	Marks
• Provides definition indicating understanding of design specifications	2
• Identifies features of design specifications	1

Sample answer/Answers could include:

Design specifications refer to a set of documentation and guidelines and principles used to develop the software. As such it specifies the requirements (from different users' point of view) as well as the functional components of the design to meet the requirements. It outlines the software components and how they interact with each other to produce the output (which satisfy the requirements) (ie requirements, software components, functions of software components, subsystem to overall system solution).

In some sense, software design specs provide some form of blue print for actual solution.

Question 22 (b) (ii)

Outcomes assessed: H4.2, H5.2

MARKING GUIDELINES

Criteria	Marks
• Provides description indicating understanding of how software design specification are used for quality assurance	2
• Provides identification of features of software design specification in the context of quality assurance	1

Sample answer/Answers could include:

Quality assurance in the context of software design specifications provides a set of methods and techniques which help to evaluate whether the developed software solution meets the requirements (originally set out to achieve).

That is, software design specs form the basis of criteria to evaluate whether the software solution is working correctly.

It also helps the evaluator to understand better that functionality of the software which in turn helps to determine the quality of software and give an assurance rating.

Question 22 (b) (iii)

Outcomes assessed: H4.1, H4.2, H6.1

MARKING GUIDELINES

Criteria	Marks
• Provides reasons indicating understanding of communication between software developers and users when developing design specification	2
• Provides identification of features of communication between software developers and users	1

Sample answer/Answers could include:

In developing a software system, the users and their requirements form the critical inputs to the software design. Hence it is necessary to communicate with the users to understand their requirements such as functionality and features required, the types and formats of outputs to be produced, how the various inputs are to be entered (eg keyboard, voice, mouse), the characteristics of the user environment such as the type of hardware, social and ethical issues such as special needs.

It is worth noting that there are different types of users (eg individuals, organizations, different roles of users) as well as future users (who may become users in the future).

Question 22 (b) (iv)

Outcomes assessed: H4.1, H4.2

MARKING GUIDELINES

Criteria	Marks
• Provides description indicating understanding of the benefits of CASE tools to provide specifications	2
• Provides identification of features of CASE tools	1

Sample answer/Answers could include:

Benefits of using CASE tools to produce specifications are:

– Representation, modification and evolution of design
– Helps to reduce development time (design diagrams)
– Helps to reason about system (understand better)
– Helps to present the design better to different types of users.

Question 22 (c) (i)

Outcomes assessed: H3.1

MARKING GUIDELINES

Criteria	Marks
• Provides, indicating good understanding of the topic, an identification of social or ethical issue AND • Discussion of implications on final design	3
• Provides, indicating understanding of the topic, an identification of social or ethical issue AND • Discussion of implications on final design	2
• Provides, indicating a limited understanding of the topic, an identification of social or ethical issue OR • Identification of features of impact on final design	1

Question 22 (c) (i) (continued)

Sample answer/Answers could include:

Ethical/social issues could include (and discuss the chosen issue)

- Users with limited computing (eg same section of population eg senior citizens)
- Users with disability (eg partially blind)
- eg Need for

  ⇒ Voice input/output system
- Uses with language (eg English) difficulty
- Privacy issues

  ⇒it contains confidential information (eg Tax)

  ⇒who gets to read the information?
- User friendly interface and clear instructions as to how to use the system.

Question 22 (c) (ii)

Outcomes assessed: H1.2, H4.1, H5.1

MARKING GUIDELINES

Criteria	Marks
• Provides discussion of the software developer's responsibilities in developing the tax return software indicating a good understanding of the issue	3
• Provides description of features of the software developer's responsibilities in developing the tax return software indicating an understanding of the issue	2
• Provides identification of a feature of the software developer's responsibility	1

Sample answer/Answers could include:

Some of the key responsibilities of the software developer in designing this software tax system include:

- Correctness and reliability of software produced
  - ⇒ Program needs to produce right answers
  - ⇒ Program needs to produce consistent answers and some answers for same data
  - ⇒ Software integrity
- Security of software
  - ⇒ Program needs to ensure that confidential and private data be protected
  - ⇒ Free from viruses, malicious Trojan horses
- Software maintenance
  - ⇒ Support must be provided to maintain and update the software
  - ⇒ Timely updates and their installation

Question 22 (c) (ii) (continued)

- Responsiveness
  ⇒ Facilities such as help desk, online help for users
  ⇒ support for tax office to install and operate this software
- Code of conduct for software developers
  - Use of systematic methodology for software
  - Provision of software documentation
  - If using other software, providing suitable acknowledgement, obtaining of permission etc.

Question 23 (a) (i)

Outcomes assessed: H5.2

MARKING GUIDELINES

Criteria	Marks
• Constructs an IPO diagram or IPO chart AND • Correctly puts together INPUTS AND OUTPUTS AND PROCESSES	3
• Constructs an IPO diagram or IPO chart AND • Correctly puts together INPUTS OR OUTPUTS OR PROCESSES	2
• Attempts to construct an IPO diagram OR IPO chart OR • Correctly puts together INPUTS OR OUTPUTS OR PROCESSES	1

Sample answer/Answers could include:

INPUT	PROCESS	OUTPUT
Rectangle Length Width	Calc Rectangle Perimeter = 2 x length 2x width Area = length * width	Perimeter = (perimeter) Area = (area)
Circle	Calc – Circ	Circle

Question 23 (a) (ii)

Outcomes assessed: H4.2, H4.3

MARKING GUIDELINES

Criteria	Marks
• Identifies module 'Calc_Circ' as a stub	1

Sample answer/Answers could include:

Calc_Circ

Question 23 (a) (iii)

Outcomes assessed: H5.2

MARKING GUIDELINES

Criteria	Marks
• Provides substantially correct pseudocode for the module	3
• Provides pseudocode for the module	2
• Attempts pseudocode for the module	1

Sample answer/Answers could include:

Function subprogram Calc_Circ (message) INPUT radius
Circumference = 2* π * radius
Area = π * radius * radius
Message = 'circumference ='; circumference
'area ='; area
RETURN message
END

Question 23 (b) (i)

Outcomes assessed: H4.2, H4.3

MARKING GUIDELINES

Criteria	Marks
• Locate and name a control structure	1

Sample answer/Answers could include:

Pre-test loop/Pre-test repetition

Question 23 (b) (ii)

Outcomes assessed: H4.2, H4.3

MARKING GUIDELINES

Criteria	Marks
• Names the search as linear search	1

Sample answer/Answers could include:

Linear search

Question 23 (b) (iii)

Outcomes assessed: H4.2, H4.3

MARKING GUIDELINES

Criteria	Marks
• Provides modified algorithm that is substantially correct	2
• Provides modified algorithm indicating understanding	1

Sample answer/Answers could include:

Between 120 and 130 Found = False
Between 160 and 170 Found = Time
Between 190 and 200 If Found = False Then

PRINT 'Target Rainfall not found'

End If

Question 23 (b) (iv)

Outcomes assessed: H4.2, H4.3

MARKING GUIDELINES

Criteria	Marks
• Explains why software developers use standard modules AND • Gives an example	3
• Indicates an understanding of the use of standard modules by software developers	2
• Provides a limited understanding of standard modules OR identifies a standard module	1

Question 23 (b) (iv) (continued)

Sample answer/Answers could include:

Standard software modules are used in the development of software solutions as they help to:

⇒ reduce software development time (on the assumption they are correct)

⇒ on the assumption that standard software modules are correctly written – they compare quality of overall solution (less or no errors)

- It is assumed that standard software modules have been tested in a variety of applications. Hence there is a greater chance of them working properly for the system being developed
- Standard software modules are easier to understand (as they would have been used before)

Examples of standard algorithms other than searching/sorting

- Standard user interface design algorithms (eg main system)
- Data conversion algorithms
- Security algorithms (eg encryption, integrity checking).

Question 23 (c) (i)

Outcomes assessed: H4.2, H4.3

MARKING GUIDELINES

Criteria	Marks
• Identifies a reason providing an understanding of syntax and structure	1

Sample answer/Answers could include:

The structure GET has a < variable list >
<Variable list> is a set of variables reported by a comma

- Variable is <letter> followed by <letter>
- Letter is A…Z and a….z

Name, number, age ⇒ is a list of variables separated by a comma

Name }
Number } are variables; ie letter followed by letter
Age }

⇒ Yes, statement is legal.

Question 23 (c) (ii)

Outcomes assessed: H4.2, H4.3

MARKING GUIDELINES

Criteria	Marks
• Provides substantially correct discussion of statement legality	2
• Identifies an issue of legality indicating understanding of syntax and structure	1

Sample answer/Answers could include:

IF number:=7 THEN name:= Tom ENDIF

Syntax not legal because

- IF – ENDIF shoud be IFF – ENDIFF
- number: = 7 is incorrect should be number = 7
- Name: = TOM is incorrect because variable has been assigned to TOM which has not been declared to be constant

Question 23 (c) (iii)

Outcomes assessed: H4.2, H4.3

MARKING GUIDELINES

Criteria	Marks
• Defines a post-test repetition using a railroad diagram	3
• Indicates an understanding of a post-test repetition using a railroad diagram	2
• Attempts a railroad diagram	1

Question 24 (a) (i)

Outcomes assessed: H1.2, H2.1

MARKING GUIDELINES

Criteria	Marks
• Defines programming paradigm	1

Sample answer/Answers could include:

A model, used in this context to refer to a type of programming language.

Question 24 (a) (ii)

Outcomes assessed: H1.2, H2.1

MARKING GUIDELINES

Criteria	Marks
• Names ONE object-oriented language	1

Sample answer/Answers could include:

C^{++}

Question 24 (a) (iii)

Outcomes assessed: H4.2

MARKING GUIDELINES

Criteria	Marks
• Provides a discussion that demonstrates a substantial understanding of the similarities and differences between the imperative and functional paradigms	3
• Indicates an understanding of the similarities and/or differences between the imperative and functional paradigms	2
• Demonstrates some knowledge of the characteristics of different programming paradigms	1

Question 24 (a) (iii) (continued)

Sample answer/Answers could include:

- Both can utilise a modular approach based on common building blocks eg functions/procedures
- Both can be used to solve a number of different types of problems, though functional languages lend themselves to mathematical problems
- Imperative programming tends to result in more complex solutions with a greater amount of coding
- Functional programming has more simplified coding and allows for greater re-use of code
- Functional programming lends itself to the processing of lists of simple data, data streams etc whilst imperative programming works easily with a broad range of data types AND data structures
- Functional programming lends itself to recursive repetition and classical problems with recursive solutions eg Towers of Hanoi.

Question 24 (b) (i)

Outcomes assessed: H2.1, H2.2

MARKING GUIDELINES

Criteria	Marks
• Demonstrates knowledge of inference engines	1

Sample answer/Answers could include:

The inference engine is the built-in processing part of a logic programming language that use the knowledge base to derive conclusions to the queries of the user.

Question 24 (b) (ii)

Outcomes assessed: H4.2

MARKING GUIDELINES

Criteria	Marks
• Demonstrates a substantial understanding of the application of facts and rules to extrapolate information	3
• Demonstrates an understanding of the use of facts and rules	2
• Demonstrates some knowledge of the use of facts or rules	1

Sample answer/Answers could include:

Natasha is a student
Robert is a student
Peter is a teacher
Natasha, Robert and Peter attend SDD classes
Robert and Natasha are SDD students
Peter teaches SDD
Peter teaches Robert and Natasha

Question 24 (c) (i)

Outcomes assessed: H4.2

MARKING GUIDELINES

Criteria	Marks
• Provides an evaluation demonstrating an understanding of the application of functions	2
• Provides an answer demonstrating a limited understanding of functions	1

Sample answer/Answers could include:

CBN ([3,9,2, AARON], [AS])
= [3,9,2, AARON, AS]
RFIND (4, [3,9,2, AARON, AS])
= 9

Question 24 (c) (ii)

Outcomes assessed: H4.2

MARKING GUIDELINES

Criteria	Marks
• Provides an evaluation demonstrating an understanding of the application of functions	2
• Provides an answer demonstrating a limited understanding of functions	1

Sample answer/Answers could include:

LFIND (3, [45, 6,2, TOM])
= 2
RFIND (2, [PINK, BLUE, RED, GREEN])
= RED

Question 24 (d) (i)

Outcomes assessed: H1.2

MARKING GUIDELINES

Criteria	Marks
• Outlines examples of different instances of the class user	2
• Identifies some features of the class	1

Sample answer/Answers could include:

Mary is an Admin user with an instance of the class user eg login_alias= ‘Mary1’, password = ‘Mary 123’ and user_level =’Admin’.

John is a client user with another/different instance of the class user eg login_alias = ‘Johno’, ‘password 1234’ with user_level’ = ‘Client’.

Question 24 (d) (ii)

Outcomes assessed: H1.2, H4.1

MARKING GUIDELINES

Criteria	Marks
• Provides sequence of processes including information relating to messages between internal and external instances	3
• Provides sequence of processes indicating understanding of class 'user' at the time of the process_login	2
• Provides sequence of processes indicating an understanding of classes	1

Sample answer/Answers could include:

– A 'long_alias' is passed to the 'process_login' method
– The 'get_password' method is called passed the 'login_alias'. It would ask the user for their password and return the password to 'process_login'
– 'Retreive_DBpassword' method is called also passed the 'long_alias' and returns the stored password for the user
– The entered password is compared to the stored password
– If they are the same then the 'valid' attribute is set to true, otherwise set to false
– This 'valid' attribute could then be used when accepting or denying the login attempt.

Question 24 (d) (iii)

Outcomes assessed: H1.2, H4.2

MARKING GUIDELINES

Criteria	Marks
• Provides explanation of specialisation and generalisation	2
• Provides description of specialisation OR generalisation	1

Sample answer/Answers could include:

A teacher is the subclass of a human being, whilst a human being is the superclass.
A subclass is a specialised version of a superclass and a superclass is a generalised version of a subclass.

Question 25 (a) (i)

Outcomes assessed: H1.1, H1.3

MARKING GUIDELINES

Criteria	Marks
• Correctly evaluates the expression showing working in binary	2
• Computation, showing the working, displaying an understanding of the procedure	1

Sample answer/Answers could include:

$10101011 \div 00001001$

$$
\begin{array}{r}
10011 \\
1001\overline{)\ 10101011} \\
-1001 \\
\hline
1101 \\
-1001 \\
\hline
1001 \\
-1001 \\
\hline
0
\end{array}
$$

Question 25 (a) (ii)

Outcomes assessed: H1.1, H1.3

MARKING GUIDELINES

Criteria	Marks
• Describes methods of representing fractions	3
• Outlines methods of representing fractions	2
• Identifies a method to represent fractions	1

Sample answer/Answers could include:

A discussion of fixed and floating point

- fixed point
  - whole number component converted to binary
  - fraction component converted to binary

eg $6 \rightarrow \frac{5}{8}$:

$6 \rightarrow 110$

$\frac{5}{8} \rightarrow \frac{1}{2} + \frac{0}{4} + \frac{1}{8}$

$1 \quad 0 \quad 1$

- floating point
  - single and double precision
  - sign bit, mantissa and exponential.

Question 25 (a) (iii)

Outcomes assessed: H1.1

MARKING GUIDELINES

Criteria	Marks
• Correctly shows how the data would be stored, including correct addressing	2
• Shows an understanding of the storage of data	1

Sample answer/Answers could include:

Address	Data
2E	11000110
2F	11001001
30	11100010
31	11001000

Question 25 (b) (i)

Outcomes assessed: H4.1

MARKING GUIDELINES

Criteria	Marks
• Provides a purpose of truth tables	1

Sample answer/Answers could include:

The purpose of a truth table is to examine the output from a circuit or logic gate.

Question 25 (b) (ii)

Outcomes assessed: H1.1, H4.1

MARKING GUIDELINES

Criteria	Marks
• Draws a truth table which is substantially correct	3
• Draws a truth table which indicates an understanding of truth tables	2
• Displays some knowledge of the components of truth tables	1

Sample answer/Answers could include:

A	B	C $\overline{A}$	D A or B	E C nor D
0	0	1	0	0
0	1	1	1	0
1	0	0	1	0
1	1	0	1	0

Question 25 (c) (i)

Outcomes assessed: H1.1, H1.3

MARKING GUIDELINES

Criteria	Marks
• Correctly identifies/calculates the contents of T	2
• Displays an understanding of the process described	1

Sample answer/Answers could include:

R	S	T
0	0	0
1	0	0
0	1	0
0	0	0
1	0	0
0	1	0
1	0	0
1	1	1

Question 25 (c) (ii)

Outcomes assessed: H1.1, H1.3

MARKING GUIDELINES

Criteria	Marks
• List the 8-bit strings showing substantial understanding	2
• Lists an 8-bit string which correctly answers the question	1

Sample answer/Answers could include:

R 10111101
τ 10111101
x x

Now

R and S $\rightarrow \tau$

So

1 and 1 $\rightarrow$ 1

but 0 and (0,1) $\rightarrow$ 0

So

S could be

10111101
10111111
11111101
or 11111111

Question 25 (d) (i)

Outcomes assessed: H1.1, H1.3, H4.1

MARKING GUIDELINES

Criteria	Marks
• Identifies elements of header information, showing substantial understanding of header contents	2
• Identifies components of data packet	1

Sample answer/Answers could include:

Headers may contain
- Start of packet
- Packet id
- Destination address
- Source address
- Error checking protocol
- Data protocol
- Port number

Question 25 (d) (ii)

Outcomes assessed: H1.1, H1.3, H4.1

MARKING GUIDELINES

Criteria	Marks
• Compares and contrasts the contents of data packets; header information, data characters and trailer information, from each of the cases	3
• Describes packets displaying an understanding of data packets	2
• Displays a limited understanding of the contents of data packets	1

Sample answer/Answers could include:

Compare
USB and scanner both:
- send packets containing header information, data characters and trailer information.
- use error checking.
- use component id.

Contrast
- Data packets are likely to be of different size (mouse – smaller; scanner – larger)
- Different error checking protocols may be used (mouse – parity; scanner – more sophisticated)
- Substantially more information in data character stream for the scanner
  - pixel information
  - colour etc
  - size

  rather than mouse
  - x/y co-ordinates (changes)
  - mouse clicks
  - mouse roller/scroll.