**THE BCS PROFESSIONAL EXAMINATION**
**Diploma**

**April 2004**

**EXAMINERS' REPORT**

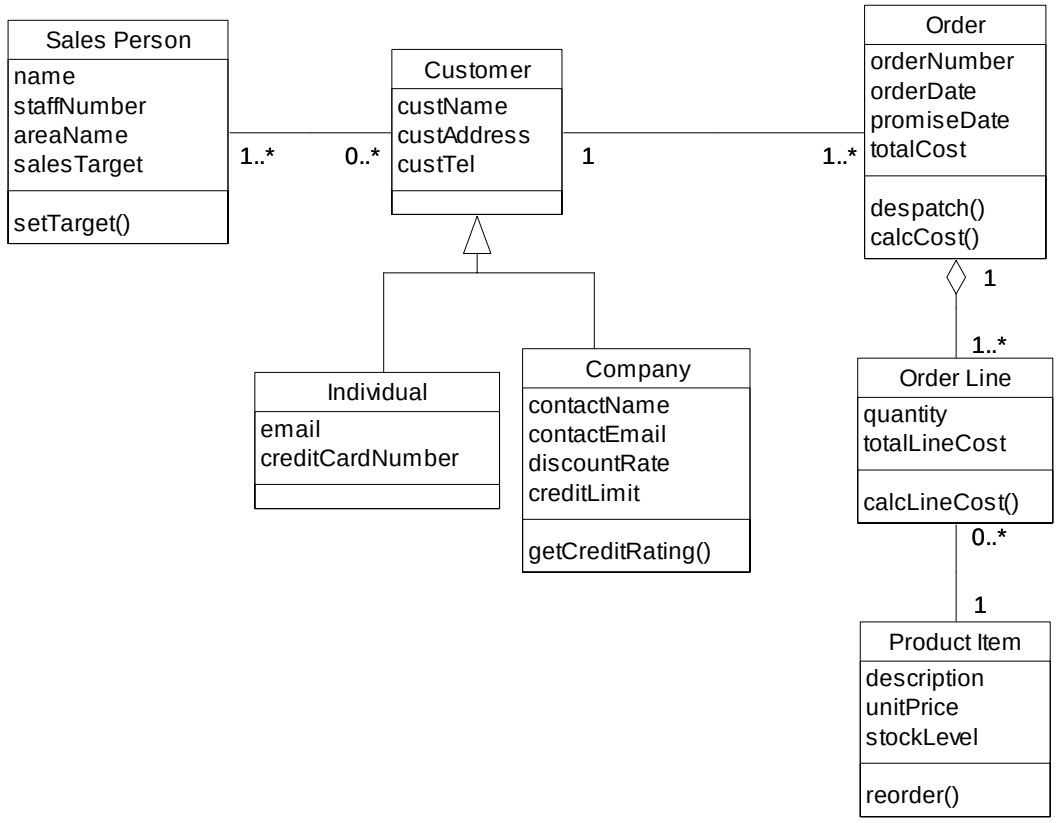
# Systems Design

## General Comments

The results of this examination were encouraging with a good overall pass rate. Candidates should however ensure that they read the question carefully and answer the question as put to ensure full marks.

## Question 1

The following class diagram fragment is part of a design for an order processing system. It is to be mapped to a Relational Database Management System (RDBMS).




*a)* Briefly explain how each of the following elements of the class diagram may be mapped:

*i)* a class, (2 marks)
*ii)* a one to many association, (2 marks)
*iii)* a many to many association, (2 marks)
*iv)* the 'Customer' inheritance hierarchy, (5 marks)
*v)* the 'Order-Order Line' aggregation. (4 marks)

*b)* Produce a suitable relational schema (set of normalized tables) for the above class diagram. (10 marks)

**Question 1 Answer pointers**

1a)(i) class

For each class create a relation with class attributes becoming columns in the proposed table, determine the primary key (if there are no suitable attributes for this create a primary key attribute).

(2 marks)

1a)(ii) one to many association

For each class create a relation as above and post the primary key from the relation created for the uni-part class (one end), into the relation created for the multi-part class (many end) to become a foreign key.

(2 marks)

1a)(iii) many to many association

For each class create a relation as above and create a new relation for the association, create the primary key as the composite of the primary keys of the relations created for the classes participating in the association.

(2 marks)

1a)(iv) the 'customer' inheritance hierarchy

Students may identify one of three strategies:

a) implement all classes as tables posting customer PK into child class tables;

b) only implement the superclass; subclasses collapse into attributes (with null values where not used);

c) only implement subclasses; superclass is redundantly copied in each subclass.

(4 marks for any suitable strategy + 1 if existence of >1 strategy is identified, total 5 marks)

1a)(v) the 'order-order line' aggregation

This aggregation is a 1 to 1..* multiplicity, therefore deal with this as per the one to many association above. (NB whatever is done with the order on the db, e.g. view, delete; must be done to the corresponding order lines)

(3 marks for strategy + 1 if significance of aggregation is identified, total 4 marks)

1(b) Produce a suitable relational schema (set of normalized tables) for the above class diagram.

(10 marks)

The following is a sample solution. Students answer may differ depending upon strategy chosen for dealing with hierarchy – marked on merit.

SALES PERSON (Staff-Number, Staff-Name, Area-Name, Sales-Target)
CUSTOMER SALES PERSON (Staff-Number, Cust-ID, transaction)

CUSTOMER (Cust-ID, Cust-Name, Cust-Addr, Cust-Tel)
COMPANY (Cust-ID, Contact-Name, Contact-Email, Discount-Rate, Credit-Limit)
INDIVIDUAL (Cust-ID, Email, Credit-Card-Number)
ORDER (Order-Number, Order-Date, Promise-Date, Total-Order-Cost, Cust-ID)
ORDER LINE (Order-Number, Product-Code, Quantity, Total-Line-Cost)
PRODUCT ITEM (Product-Code, Product-Description, Unit-Price, Stock-Level)

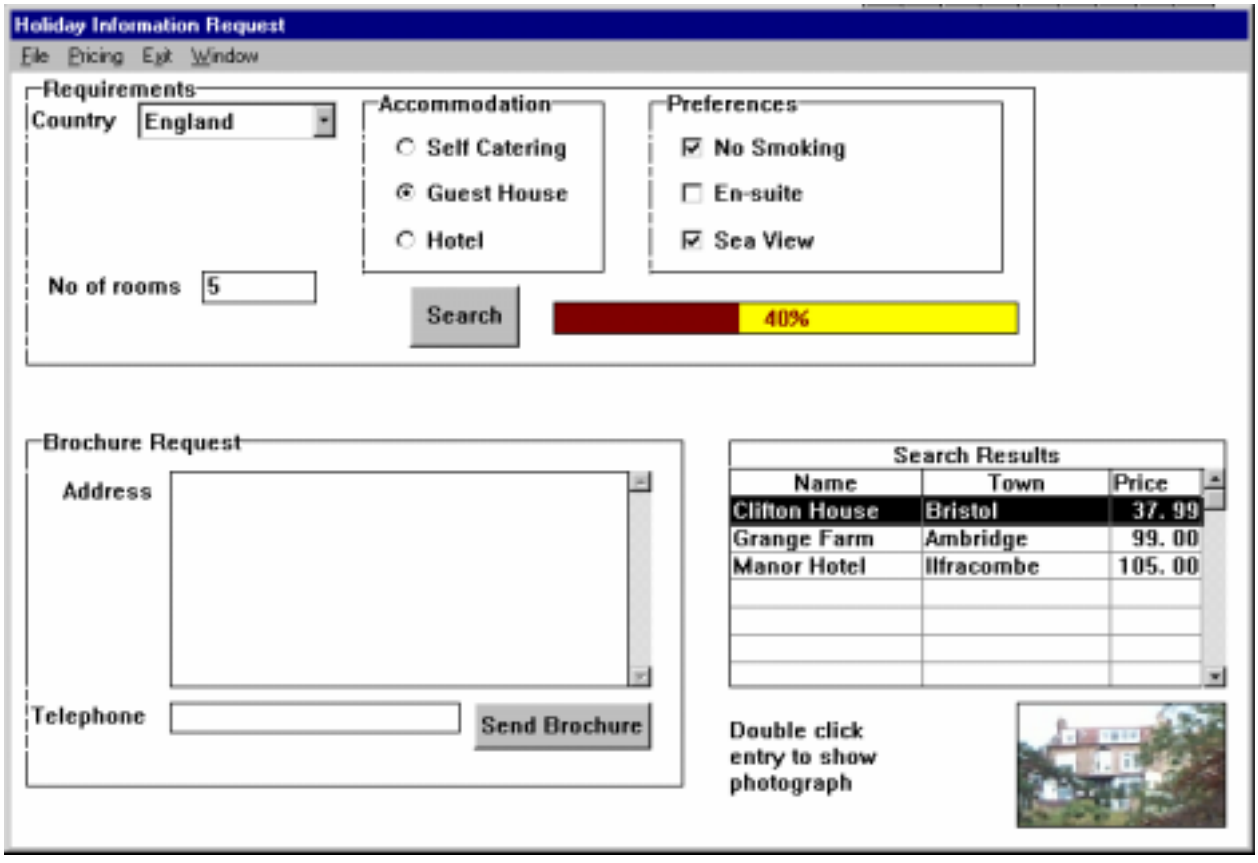
Identification of tables = 4 marks (adjusted according to chosen strategies in (a))
Suitable use of PKs & FKs = 3 marks
Assignment of attributes = 3 marks (total 10 marks)

**Examiner's Comments**
This question was popular with most candidates attempting it. Of those who did it, approximately 67% achieved a pass mark for it and there were some very good answers. Other answers could have been improved as some candidates for part (a) replicated answer pointers from previous papers without adding any further detail.

In part (a) weak candidates did not cover how tables in a RDBMS could be designed to store objects from the classes represented in class diagram, but basically described *what* a class, a 1 to 1..* association, aggregation etc. are which is not what the question was asking. A small number of candidates misunderstood the term mapping, incorrectly thinking it to be how a class etc. was represented (i.e. what it looked like when drawn) in the diagram.

Part (b) was fairly well answered, although some candidates did not always apply the principles covered in part (a) to the actual production of the normalized tables in (b). Consequently, the CUSTOMER SALES PERSON table was frequently missed and the strategies for dealing with inheritance and aggregation were ignored.

**Question 2**
The GUI window illustrated below contains various different components, including Menus, Drop-down list, Radio-buttons, Check boxes, Command Buttons, Text and Number Fields, Groups, a Picture and a Thermometer.




Draw an object class (inheritance) hierarchy showing the various component classes illustrated.

You should use either the general format illustrated below or another notation with which you ***are familiar.***

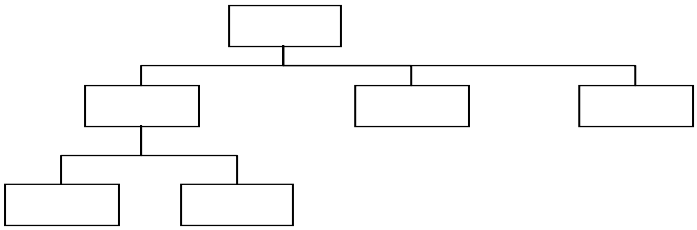
Each box should show the class name and one or more attributes appropriate to that class.

The top class should be called “all screen objects” and show some attributes appropriate to all screen objects.

You should show AT LEAST NINE different classes (9 marks)

AT LEAST EIGHT Subclass:Superclass relations (8 marks)

and AT LEAST EIGHT different attributes. (8 marks)



**Question 2 Answer Pointers**

This question aims to test knowledge of the different screen objects available to designers, and the ability to categorise them, as well as some understanding of Object Oriented thinking.

Clearly there are different ways screen objects can be categorised, so marks were awarded for any sensible and/or useful arrangements.

A possible arrangement (using indentation to signify the next level in the hierarchy, with attributes in brackets):

Screen object (protected, visible, colour, x, y, height, width)
- Container object (label)
  - Window (name, resizable)
  - Group (name)
  - Radio Button Set
- Command Button (label or graphic)
- Field
  - Text field (font)
    - Single line
    - Multi-line
    - Dropdown (options)
  - Number (format)
- Check box (default)
- Radio Button
- Table (rows, cols)
- Picture (bitmap)
- Menu
- Thermometer (value)

Marks: Each subclass – 1 mark to maximum of 9

Each valid superclass:subclass relation – 1 mark to maximum of 8

Each valid attribute – 1 mark to maximum of 8

Marks allowed for partial misread:

Attributes and subclasses correct, but no inheritance – max 15 marks

Class hierarchy of application (rather than screen) objects – max 8 marks

Containment hierarchy of application objects – max 5 marks

**Examiner's Comments**
The question was not popular, being attempted by less than half the candidates. Many of them lost marks by failing to realise that they were being asked for an **inheritance** diagram of **screen** objects. Unfortunately this was fundamental to the question, and without it most of the available marks were lost. Some gave containment hierarchies of application objects, some gave database diagrams, and some were a bit muddled and fell between several stools.

The problem appears to be that some candidates jump to conclusions about what they are being asked for, and then fail to read the question carefully enough to confirm or deny those assumptions.

The majority of the candidates who understood the question correctly did very well, showing sensible superclass:subclass relations as well as appropriate attributes, and consequently gained high marks.

**Question 3**
Write a BRIEF explanation of FIVE of the following terms as used in Object-Oriented systems design. You should illustrate your answer with an example and/or diagram, as appropriate:

| | | |
|---|---|---|
| *i)* | encapsulation, | (5 marks) |
| *ii)* | sequence diagram, | (5 marks) |
| *iii)* | software reuse, | (5 marks) |
| *iv)* | package diagram, | (5 marks) |
| *v)* | statechart (state diagram), | (5 marks) |
| *vi)* | coupling and cohesion, | (5 marks) |
| *vii)* | deployment diagram. | (5 marks) |

(5 x 5 marks, total 25 marks)

**Question 3 Answer pointers**
The candidate's explanation should include basic overview of each term covering such points as outlined below:

i) encapsulation;
- the grouping of operations and attributes into an object structure
- the operations provide the sole facility for the access or modification of the attributes' values
- use of private/public/protected interfaces
- encapsulation may also be applied to packages
- each object (or package) must have a well defined interface to allow services to be invoked

ii) sequence diagram;
- dynamic view of the system
- usually drawn for a single use case
- shows interactions (message passing) between objects in a time sequence
- passage of time represented in sequence diagrams linked with processes (message passing)

iii) software reuse;
- use of classes of objects in different systems/sub-systems
- usually groups of objects (components) or even entire subsystems are reused
- encapsulation important in achieving reuse
- low interaction coupling promotes reuse
- need to develop and 'advertise' well documented class libraries in order to achieve a high level of reuse

iv) package diagram;

- shows packages (groupings) of classes
- also shows the dependencies among them
- used to show software architecture in a logical way
- packages can be nested within other packages

v) statechart (state diagram);

- dynamic view of the system
- is drawn for any single class of objects with non-trivial behaviour
- shows all events affecting that class & all its possible states
- state is determined by the value of an objects attributes and its associations

vi) coupling and cohesion;

- coupling is the degree of interconnectedness of different elements of the System
- i.e. the degree of interaction an object has with other objects
- low interaction coupling and high inheritance coupling are desirable
- can be applied to objects or components (compiled packages)
- cohesion is the degree to which an element of the system is focused on a single purpose
- high cohesion is desirable
- can have class cohesion, operation cohesion and specialisation cohesion

vii) deployment diagram.

- shows the physical relationships among software & hardware components
- shows how packages/components are implemented on hardware platforms
- can show particular 'named' machines
- show communication links and protocols between machines

Alternative explanations and diagrams/examples provided will be marked on merit.

(5 x 5 marks, total 25 marks)

**Examiner's Comments**

This question was also popular with most candidates attempting it. Of those who did it, approximately 70% achieved a pass mark for it and the question was generally answered quite well. Many candidates provided clear and appropriate annotated diagrams to illustrate their explanations and obtained good marks. Of the candidates that scored poorly on this question, most did not provide examples and/or diagrams even though the question clearly asks for them.

Several candidates demonstrated common misunderstandings, for example the question explicitly states that the candidate should explain the terms as used in *object-oriented* systems development. Frequently candidates discussed terms as applied to *structured* systems development. For instance, many candidates described coupling and cohesion in terms of structure charts and modules and not in terms of objects; i.e. explaining coincidental, logical and temporal cohesion as opposed to class, operation and specialisation cohesion.

Additionally, when discussing software reuse some candidates covered use of 4GLs to achieve reuse and not the class and component reuse of object-oriented systems. Also some candidates lost marks by only explaining what software reuse is and didn't then cover how it may be achieved.

## Question 4

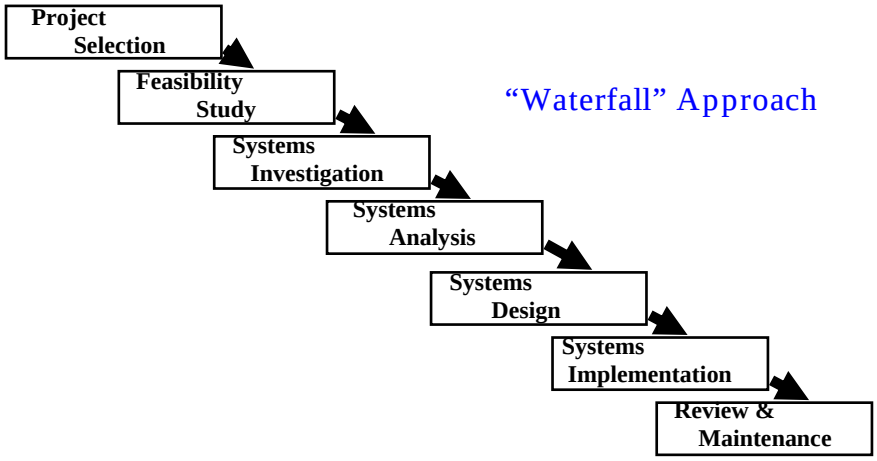
*a)* With the aid of a diagram, briefly describe each of the following System Development Life Cycles (SDLCs):

*i)* The Traditional Waterfall Approach,
*ii)* Rapid Application Development (RAD). (15 marks)

*b)* Outline the way in which logical and physical design activities are organised within each of the two above SDLCs. (10 marks)

## Question 4 Answer pointers

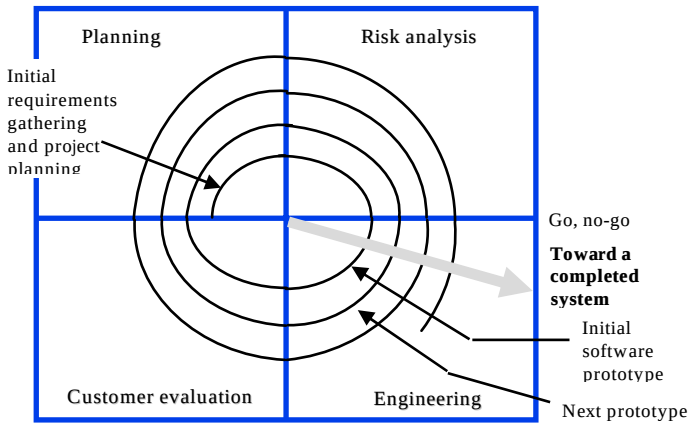
a) i) The Traditional Waterfall Approach diagram should show phases such as:




A ***brief*** statement on each of the phases may be given. Candidates may also mention the following points for which credit will be given:-

- each phase follows on after the completion of the previous phase
- iterations are possible but any that are made are very difficult and expensive
- has well defined phases
- suited to systems where requirements are well understood
- use of this approach may lead to requirements drift
- testing (verification) done late in the development process

ii) A typical RAD lifecycle is (a prototyping LC will also be acceptable):



Candidates may identify the following features of a RAD approach:-

- system divided into appropriate 'mini-projects'
- developed incrementally (frequent software releases) through iterations
- later iterations build upon previous software builds
- RAD reduces possibility of requirements drift
- development team can learn/build experience from previous iterations
- timeboxing is used to control software releases

Marks will be awarded on merit for quality of diagrams and/or descriptions given.

(15 marks)

**4.** *b)* Outline the way in which logical and physical design activities are organised within each of the two above SDLCs. (10 marks)

Candidates should base their answer around the following (suitable alternative points accepted):

- traditional SDLC all of the design is done in one phase after analysis & before implementation,
- in a RAD approach the design is undertaken in much smaller iterative phases
- in a traditional SDLC hardware/software architectures can be designed as a 'master plan' but in RAD approach base-line architectures are evolved
- in RAD the logical design is strongly influenced by previous physical design iterations
- similarly for traditional SDLC database design is done as one activity, in RAD need to merge logical/physical schemas as development progresses
- in RAD process design is often easier to understand as developing in smaller chunks, but can be more difficult to design the relationships among process (interactions)

(10 marks)

**Examiner's Comments**

This question was attempted by roughly 75% of candidate of which approximately 65% achieved a pass mark. The range in the standard of answers was quite wide with some very good answers but also some that were poor.

Part (a) was generally answered quite well with most students being able to describe the given lifecycles. Some students spent an inordinate amount of time covering the Traditional Waterfall Approach and wrote far too much for the number of marks available for this (i.e. half of the 15 marks of part (a)). The RAD lifecycle was generally covered less well.

Overall, part (b) was answered much less well and many students lost marks for failing to answer the question that had been asked. A common mistake was to discuss *what* logical and physical design activities take place rather than *how they are organised* within the two lifecycles (c.f. marking guidelines above).

## Question 5

*a)* Describe FIVE different mechanisms by which a user can navigate from one window to another within a GUI based system. (5 x 2 marks)

*b)* Briefly explain the issues to be considered when designing the navigation between windows in a multi-window interface. (15 marks)

## Question 5 Answer pointers

(a) Any sensible answers, eg. Button click; menu selection; validation error; timeout; request for help
Key press: Enter, Function Key, Hotkey
(2 marks per mechanism)

(b) Minimise keystrokes/mouse moves

Provide clear titles on windows.

Menus/button texts should be:

Meaningful

Concise

Consistent

Users shouldn't be surprised by unexpected changes

Take care about multiple windows – these can help or confuse user;
it may help if only one window is enabled at a time.

Consistency between windows

Title, Navigation buttons, Menus, general look and feel.
(but not so much consistency that they are indistinguishable!)

"Where am I?" feature

Help feature

A good range of issues should be mentioned, with evidence of understanding.
Award 1 mark for a mention, 2 if explained, 3 if well understood, to a max of 15.

Marks can therefore be earned from a few well explained issues, or from many brief mentions.

## Examiner's Comments

This was quite a popular question, and most candidates showed a good understanding of at least some of the mechanisms and issues. Part (a) was generally quite well done, but part (b) rather sketchily.

Most candidates were able to identify a few different ways a user might navigate to a new window, but most also omitted some they should have thought of.

Similarly most candidates were aware of some of the design issues and principles but failed to think as widely as they could.

Common causes for lost marks were:

Repetition, giving essentially the same answer in two ways.
Talking about general design rather than Navigation.
Thinking about a limited set of systems, e.g. Web or MS Office.

Forgetting some of the different things that can cause new windows to be loaded, e.g. errors, timeouts, keystrokes etc.

**Question 6**

a) Briefly explain FIVE different types of on-line help that may be provided within a system. (5 x 2 marks)

b) Explain FIVE principles that should be applied to the design of effective on-line help. Include both general design considerations, as well as those specific to a 'help' system. (5 x 3 marks)

**Question 6 Answer pointers**

a) Any valid types accepted. For example:
Error messages on validation,
help on help,
help on concepts,
help on procedures (how do I…?)
What does this field do?
Step-by-step guidance; 'show me'… (up to 5 x 2 marks)

b) These may be general report or screen design principles, or those specific to help systems.
At least 9 marks of this must be for Help-specific principles
<u>general</u>
Can include suitable choice of fonts, colour, layout, use of tables etc. for readability
clear navigation, so user doesn't get lost
What would have to change for a help screen to be printed?
<u>specific</u>
Simplicity, organise and show (from Hoffer, George and Valacich), or equivalent:
**Simplicity**: use short simple wording, common spelling and complete sentences
Give users only what they need to know, with ability to find additional information.
**Organise**: Use lists to break information into manageable pieces.
**Show**: Provide examples of proper use and the outcomes of such use.
The user is having difficulty, so be careful with terminology, ensure the explanation is meaningful to novice or casual user;
Provide Active help: "if you want to do this, click here".
Provide Context-sensitive help.
Structure the help as a hypertext document, allowing user to jump around.
Consider use of "Help compiler"
(Award 3 marks for each valid principle up to max 15)

**Examiner's Comments**

This question was not very popular, but on average the candidates that attempted it did reasonable work, and some produced very good answers.

They did seem aware of issues like level of user understanding and design considerations for on-screen documents, and many were aware of specific issues for Help Systems, however some candidates interpreted "help" as any IT support for a user task.

Some lost marks by describing documentation, email, telephone, support web-sites, chat-rooms etc., and missed the real points about on-line help.

Others lost marks by repeating the same point, or a very similar one, in different words.