Abstraction

Name: _____

Question 1 *[1 Mark]*

Why would this map of the Melbourne rail network be considered an abstraction (choose the most correct answer)?

(a). The map provides a broad aerial view of the city rail network

> No - while this may encode additional information, this is not an abstraction

(b). With respect to the context, the map suppresses less important details - such as the names of roads ✓

> Correct - less important details are suppressed and it is therefore an abstraction. Replacing the actual locations of the rail lines with simplified / less accurate lines in order to emphasise each line's important aspects is another example.

(c). The map shows not only the relative location of the stations but also the connectivity of the rail network

> No - these characteristics are not related to abstraction, only to a rail network

(d). The map provides additional information an aerial photograph wouldn't provide by indicating where rail lines intersect with each other

> No - while this may provide additional information, this is not an abstraction

Question 2 *[1 Mark]*

With respect to the concept of abstraction, which of the following statements is not true?

(a). A city rail map is an abstraction of the city's rail system

> True - a city rail map suppresses the unnecessary details / complexity of the real rail system

(b). A model airplane is an abstraction of the plane it represents

> True - a model airplane suppresses the unnecessary details / complexity around the actual plane - focusing only on it's appearance

(c). A city skyscraper is an abstraction of the building's floor plan ✓

> False.
>
> A city skyscraper is not a version of it's floor plan with certain details suppressed for a given context. However, conversely, it could be said that the building's floor plan is an abstraction of the building itself

(d). A mathematical function describing only the height of a bouncing ball is an abstraction of the ball's bouncing motion

> True - a mathematical function modelling the bouncing ball suppresses the unnecessary details / complexity of the ball bouncing (e.g. how it might reflect light, effects of friction etc.)

Question 3 *[4 Marks]*

Outline two ways in which an operating system hides the complexity of the hardware from users and applications.

> Award [1] mark for identifying a method of OS abstraction.
> Award [1] mark for an expansion on the first identified method.
>
> Award [1] mark for identifying a second method of OS abstraction.
> Award [1] mark for an expansion on the second identified method.
>
> FOR EXAMPLE:
>
> OS' provide a Graphical User Interface (GUI) for hiding the complexity of hardware functions;
> For example, dragging a file into a folder abstracts a series of complex hardware actions controlled by low level binary commands which moves / remaps storage locations;
>
> OS' provide a Command Line Interface (CLI) for hiding the complexity of hardware functions;
> For example, renaming a file using a rename command abstracts a series of complex hardware actions controlled by low level binary commands which check for illegal characters and store individual characters in a standard binary format, such as ASCII or UNICODE;
>
> OS' provide drive letter / label representations for hiding the complexity of hardware allocations;
> For example, simply labelling a new drive partition on secondary storage with a name / letter abstracts the complex process of allocating a new storage section and restricting access to that section to only authorised users;
>
> OS' provide applications with a set of services to access hardware functionality without having to know details about the hardware itself;
> For example, the OS could expose a "take photograph" function which takes a photograph from the device camera. The application doesn't need to know the details around the type of camera or the set of steps to establish permissions to use the camera - the complexity is hidden / abstracted so that it is easier for the application developer to create software to work on the OS;

Question 4 *[4 Marks]*

Outline two benefits of high level languages over low level languages such as Assembly.

> Award [1] mark for identifying a valid benefit.
> Award [1] mark for a valid and distinct and related expansion / elaboration related to the identified benefit.
>
> Award [1] mark for identifying a second valid benefit.
> Award [1] mark for a valid and distinct expansion / elaboration related to the identified benefit
>
> For example:
>
> ABSTRACTION:
> High level languages provide a higher level of abstraction than machine languages;
> which in turn improves MAINTAINABILITY / EXTENSIBILITY / REUSABILITY / TESTABILITY etc;
>
> PRODUCTIVITY / EASE OF USE:
> High level languages are typically more programmer friendly;
> which better supports productivity and makes code generally easier to read / write / develop etc;
>
> PORTABILITY:
> Higher level languages are machine independent;
> which means that the same code can be written to work across different machine architectures and / or operating systems;
>
> LEARNABILITY:
> Higher level language abstractions can help to support learning new programming languages;
> by supporting programmers with more human readable syntax as well as more intuitive / relatable semantics;
>
> TROUBLESHOOTING:
> Higher level languages are easier to debug / find errors;
> owing to helpful higher level abstractions which help programmers analyse / trace / reason about the code;
> and catch syntax or logical errors at compile and / or runtime;
>
> SECURITY:
> Higher level languages support better security;
> by distancing / restricting programmers from powerful low level functions which may be more effectively misused to exploit hidden / fundamental system vulnerabilities;