



Edith Cowan University
2023 ATAR Revision Seminar

ATAR Computer Science
Curriculum Dot points
Examination and study tips
Revision notes Examination questions
Examination marker comments

Prepared and presented by
Chris Anderson

Contents

System Analysis	5
System Development Methodologies	5
Key Points	5
System Development Life Cycle (SDLC)	5
Stages of the SDLC	6
Project management	7
Gantt Charts	7
PERT charts	7
Exam Notes	7
Sample Charts	8
Exercises	9
Development Documentation	11
Modelling the System	11
Context Diagrams	12
Data Flow Diagrams	13
Sample Diagrams	14
Exercises	16
System Architecture	20
Fetch-Execute Cycle	20
Key points	20
Components of the CPU	20
Stages of the Fetch-Execute Cycle	21
Virtualisation	22
Key Points	22
Cloud Computing	23
Service Models	23
Deployment Models	23
Advantages/Disadvantages of Cloud Computing	23
Disaster Recovery	24
Key Points	24
Disaster Recovery Tools	24
Managing Data	26
Database Concepts	26
Data Integrity:	26
Exercise	27

Normalisation	28
Data Anomalies.....	28
Normalisation to 3 rd Normal Form	28
Normalisation Example	30
Exercises	33
Entity Relationship Diagrams	36
Symbols.....	36
Creating an ERD.....	36
Many-to-many relationships	37
Sample ERD.....	39
Exercises	40
Legal/Ethical Issues.....	45
Exercise.....	45
Programming	47
Key Concepts	47
Data Types	47
Programming Concepts	48
Pseudocode	49
Common Commands	50
Control Structures	51
Modularisation	52
Exercises	54
Desk Checking.....	57
Completing a Trace Table.....	57
Example	57
Exercises	57
Structure Charts.....	60
Example 1	60
Example 2	61
Exercise.....	63
Networks and Communication.....	65
Terminology.....	65
Network Devices.....	65
Device Roles.....	65
Transmission Media.....	66
Wireless Transmission Media.....	66
Wired Transmission Media.....	67
Wired vs Wireless	67

Exercises	67
TCP/IP Model.....	68
TCP/IP Layers	68
Exercises	70
Network Security.....	72
Security methods.....	72
Network Threats.....	73
Exercises	73
Network Diagrams.....	75
CISCO Symbols.....	75
Example	76
Exercises	77
Exam Technique	81
Sitting the Exam.....	81
Reading time.....	81
Start paper.....	81
Preparation.....	81

System Analysis

System Development Methodologies

- types of system development methodologies
 - linear – waterfall/cascade
 - iterative – rapid application development (RAD)
 - advantages and disadvantages of linear and iterative system development methodologies

Key Points

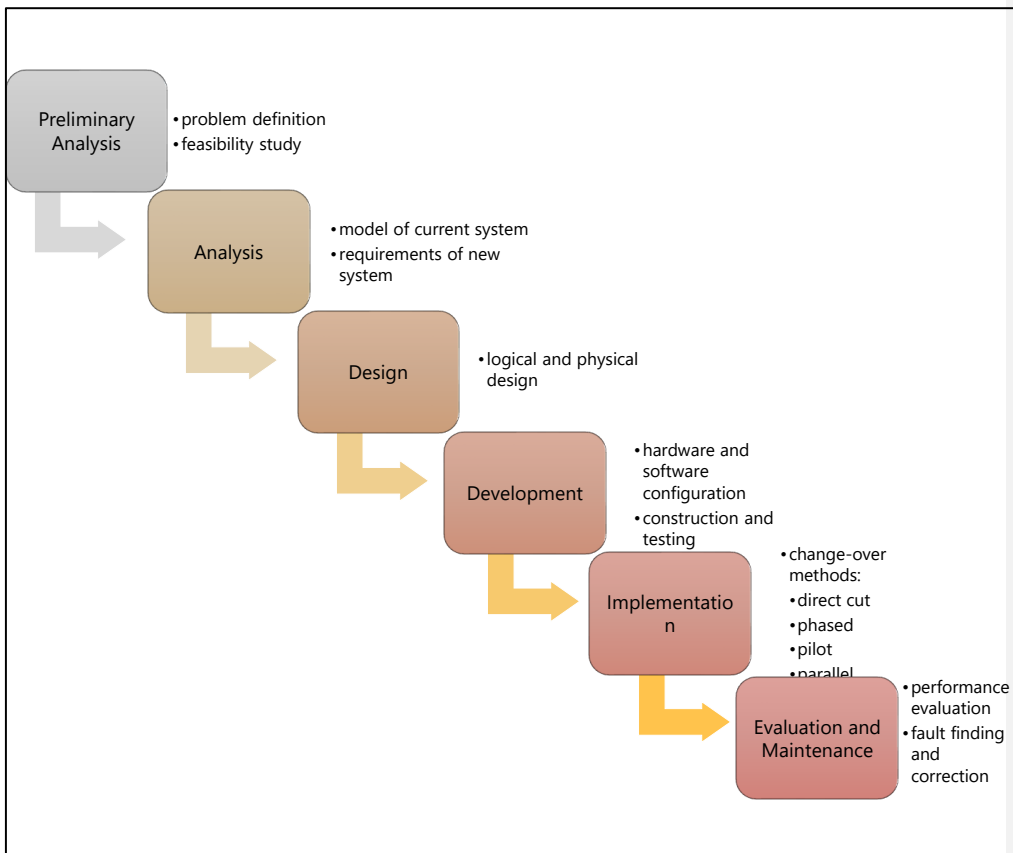
- a system development methodology is a framework that can be used to structure, plan and control the process of developing a system.
- **Linear:** (also known as waterfall or cascade) each stage is completed before the next stage starts.
 - Highly structured
 - Good for large projects
 - Less flexible
 - Easier to follow for less experienced developers
- **Iterative:** some stages (usually design and development) are repeated several times.
 - More flexible
 - Greater involvement from the end user/client
 - Allows for changes to be made to original design
 - Good for smaller projects
 - Requires highly skilled staff
 - Requires regular input from clients

System Development Life Cycle (SDLC)

- stages of the system development life cycle (SDLC)
 - preliminary analysis
 - problem definition
 - feasibility study
 - analysis
 - model of current system
 - requirements of new system
 - design
 - logical and physical design
 - development
 - hardware and software acquisition
 - construction and testing
 - implementation
 - change-over methods, including: direct cut, phased, pilot and parallel
 - evaluation and maintenance
 - performance evaluation
 - fault finding and correction
- data gathering techniques used in the SDLC, including: observation, questionnaire, interview, sample forms, and sampling volume of work processed by system

- Linear approach to development
- Underpins the structure of the Computer Science course – all the content can be considered to fit into an aspect of the SDLC
- **Use the stages named in the ATAR syllabus.** NOTE: You will see it documented differently in just about every textbook you look at!

Stages of the SDLC



Project management

- project management computer aided software engineering (CASE) tools
 - Gantt charts
 - program evaluation review technique (PERT) charts
- apply data gathering techniques and CASE tools
- analyse user and system documentation, including: Gantt charts, PERT charts

Gantt Charts

- Horizontal bar chart showing when individual tasks on a project need to be completed.
- Use arrows to show dependencies between tasks
- Key terms:
 - Dependency – a task that is dependent on a previous task being completed. That is, it cannot be started until the previous task is completed
 - Predecessor – a task that needs to be completed before the dependent task can be started
 - Concurrent – tasks that can be worked on at the same time

PERT charts

- Program Evaluation Review Technique charts
- Graphic illustration representing tasks as a network diagram
- WA ATAR exams use the Activity on Node (AON) style of PERT chart, where the activity is labelled on the node, and the time taken is on the arrow that **precedes** the node.
- The first node should be empty (or 0 if tasks are numbered)
- Key terms:
 - Critical path – the shortest possible time that the project can be finished. It is calculated from the sequence of tasks from beginning to end that will take the longest time to complete
 - Slack time – the time a task can be extended without impacting on the other tasks (that is, extending the Critical Path)
 - Lead time – time before a task can be started
 - Critical task – any task that is on the critical path. Any changes to this task will affect the critical path of the project.

Exam Notes

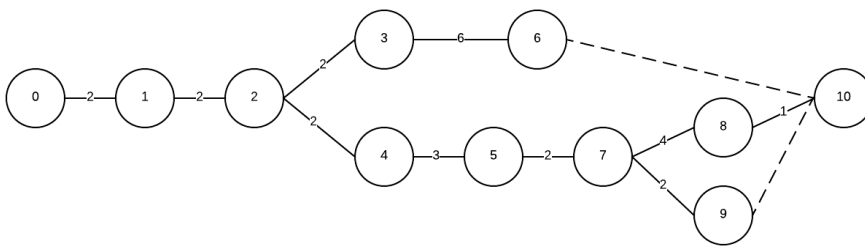
- Often in exams, but not always
- Generally easy marks if prepared for these questions
- Practice interpreting charts, in particular calculating total project time and impacts of tasks being delayed
- Practice drawing charts from table of tasks
- Practice converting between Gantt and PERT charts

Sample Charts

Consider the project management plan below.

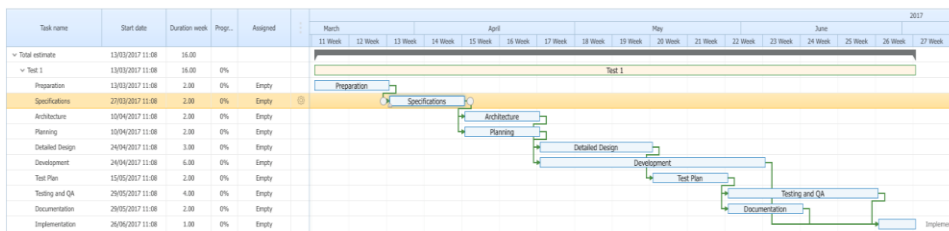
Task	Name	Duration	Team
1	Preparation	2 weeks	Marketing
2	Specifications	2 weeks	Marketing
3	Architecture	2 weeks	Planning
4	Planning	2 weeks	Planning
5	Detailed Design	3 weeks	Planning
6	Development	6 weeks	Development
7	Test Plan	2 weeks	Planning
8	Testing and QA	4 weeks	Development
9	Documentation	2 weeks	Marketing
10	Implementation	1 week	Development

PERT Chart



- (a) Identify the critical path for the current project.
1 – 2 – 4 – 5 – 7 – 8 – 10
- (b) How long will the project take to complete assuming all tasks run on time?
16 weeks

Gantt Chart



Exercises

Question 1

Jane has been hired to develop a new online sales system for RedTree, an online trading platform for people to buy and sell used goods. She has developed the following timeline for the tasks that need to be completed.

Task ID	Task Name	Duration – days	Dependency
1	Interview management	3 days	
2	Develop Context Diagram	1 day	1
3	Develop DFD	1 day	2
4	Develop un-normalised data sheet	1 day	1
5	Normalise data to 3NF	1 day	4
6	Develop ERD	1 day	5
7	Interview Salespeople	1 day	1
8	Develop Application	4 days	7
9	Develop Database	2 days	3,6, 8
10	Install Hardware	2 days	9
11	Install Application and Database	2 days	10
12	Test	1 day	11

a) Draw a Gantt chart for this project based on the information in the table above.

b) Draw a PERT chart for this project based on the information in the table above.

c) How long will it take to complete this project?

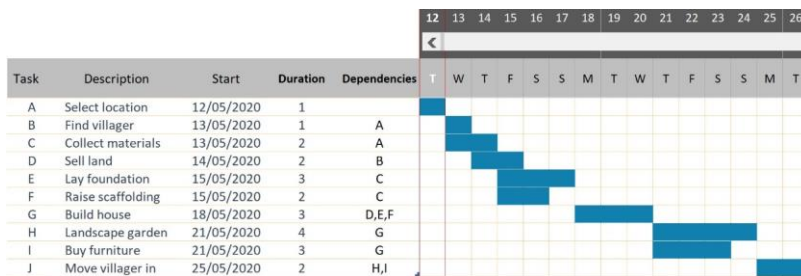
d) What is the critical path for this project?

e) The system is more complex than Jane anticipated, and it has taken her 3 days to develop a DFD. How will this affect the critical path of the project?

Question 2

Elisabeth loves her new game "Animal Junction" in which she must complete tasks to help villagers move to her virtual island. She is so obsessed with collecting new villagers that she has created a Gantt chart to help her complete the tasks as efficiently as possible.

Refer to the Gantt chart below to answer all parts of this question.



(a) Convert the above Gantt chart into a PERT chart. Draw it in the space below.

(b) Identify the critical path for this project.

(c) Explain how each of the following would affect the completion time for the project.

Laying the foundation takes 1 additional day. _____

Finding a villager takes 2 additional days. _____

Development Documentation

- systems development documentation as a part of the SDLC
 - context diagrams using Yourdon/DeMarco notation
 - data flow diagrams using Yourdon/DeMarco notation
 - system manuals
 - user manuals
- analyse user and system documentation, including: context and data flow diagrams
- create user and system documentation as a part of the SDLC
- apply context diagrams and data flow diagrams, using Yourdon/DeMarco notation, as a part of the SDLC
 - detect errors in diagrams
 - define system boundaries
 - create accurate diagrams
 - create context diagrams
 - create Level 0 DFDs
 - create Level 1 DFDs


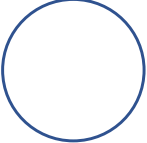

Modelling the System

- Used to model a system: both the current system and new system
- Based on data that has been gathered by the system analyst
- Provide graphical representation of what is happening in the system
- Useful to make sure that we understand what is happening in the system and what any development project needs to be able to do
- Generally used in the Analysis stage of the SDLC to analyse the current system and derive a set of system requirements for a new system
- Can also be used in the Design stage to help design a new system

Context Diagrams

- Show how the system interacts with external entities and what data is passed in and out of the system
- Shows the boundaries of the system

Symbols


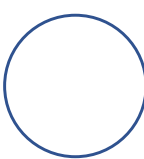


Symbol	Name	Description
	Data flow	Indicates the flow of data into or out of the system: <ul style="list-style-type: none"> • Must be labelled • Must flow between an external entity and the system • Must refer to the data, not material things • Good rule of thumb is to add <i>details</i> to the end of all data flows
	System	Represents the system <ul style="list-style-type: none"> • Must always be named • Name must always include the word System (NOTE: You will lose marks if you do not include the word System in the name)
	External Entity	Provides input to or receives output from the system <ul style="list-style-type: none"> • Should be labelled using a singular noun • Can be: <ul style="list-style-type: none"> ○ A sink: receives data from the system ○ A source: provides data to the system ○ Both a sink and a source • Must have at least one data flow, either in or out of the system

Steps to Create a Context Diagram

1. Read the scenario!
2. Identify the system: draw a circle around the system name if appropriate
3. Identify any external entities: draw a box around all external entities
4. Identify any data that flows between the system and the external entities: underline the data
5. Draw a circle and label the system
6. Draw the external entities around the system
7. Add the data flows between the external entities and the system

Data Flow Diagrams

Symbols

Symbol	Name	Description
	Data flow	Indicates the flow of data into or out of the system: <ul style="list-style-type: none"> • Must be labelled • At least one end of a data flow must always connect to a process • Must refer to data, not material things • Good rule of thumb is to add <i>details</i> to the end of all data flows
	Process	Transforms data within the system <ul style="list-style-type: none"> • Must always be labelled with a number (such as 1.0, 2.0 etc) • Number does not have to be in order (although this does make it easier to read your diagram) • Name must be some form of verb describing what the process does (e.g. Process new member) • Must transform data in some way (i.e., data flows in should be different to data flows out) • Must have at least one data flow in • Must have at least one data flow out
	External Entity	Provides input to or receives output from the system <ul style="list-style-type: none"> • Should be same external entities as the context diagram • Data flows in and out should match context diagram • Data flows must connect to a process (i.e., cannot have data flow between two external entities or from an external entity to a data store)
	Data Store	Stores data within the system <ul style="list-style-type: none"> • Must not transform data • Usually have at least one data flow in and at least one data flow out • Should be named with a noun that reflects the type of data being stored.

Steps to Create a Level 0 Data Flow Diagram

1. Read the scenario!
2. Identify any external entities: draw a box around all external entities
3. Identify the data stores: draw an open-ended box around them
4. Identify any data that flows between the system and the external entities: underline the data
5. Identify any data that flows in or out of the data stores: underline the data
6. Identify any processes that act on data: draw a circle around these
7. Identify any data associated with these processes: underline the data
8. Draw circles for the processes that you have identified
9. Add the external entities (should be the same as the Context Diagram) and data stores that you have identified
10. Add the necessary data flows
11. Check that your diagram matches the Context Diagram. The DFD should have the same external entities and the same data flows in and out of the system

Level 1 Data Flow Diagrams

- To make a DFD readable, we need to limit the number of processes to 7-8 processes
- With a complex system, we will need to break some of the Level 0 processes down into more detail
- Levelling a DFD allows us to drill down into a process to see what is happening in that process
- Ideally, we should keep levelling our DFD until each process has one data flow in and one data flow out
- Process to create a Level 1 DFD is the same as Level 0 DFD except:
 - Processes should be numbered according to the corresponding process in the L0 DFD. For example, when levelling process 3 from the L0 DFD, processes should be numbered 3.1, 3.2, 3.3 etc
 - A Level 1 DFD **does not include External Entities**
 - There will be some data flows that appear to flow to nothing – these should connect with another process or an External Entity in the Level 0 DFD
 - The Level 1 DFD should match the corresponding process in the L0 DFD

Sample Diagrams

Scenario

OutSourceIT is an online company that has been set up to help businesses with freelance developers. Businesses can post jobs on the website that they need completed and registered developers can browse the jobs and offer their services for any jobs that they would like to complete.

To help streamline the process, OutSourceIT has developed a new app called OutSourceForge that will allow businesses to post jobs that they wish to outsource to freelance developers. Companies must first register their details with the system which are then stored in the database. Once registered, they will receive login details which will allow them to access the system and submit jobs that they need completed. As part of this process, they submit a job description, including details of when the job needs to be completed, the languages to be used and platforms that the job will be required to run on. The job information is stored in the jobs file.

Freelance developers can join the OutSourceIT freelance network by paying an annual fee. When they sign up, they fill in a form listing their skills and experience as well as providing a brief personal summary. This information is then stored in the Developers table so that they can be matched to potential jobs and companies are able to browse their details. Once their information has been entered, each freelancer is sent a confirmation email with their personal login details.

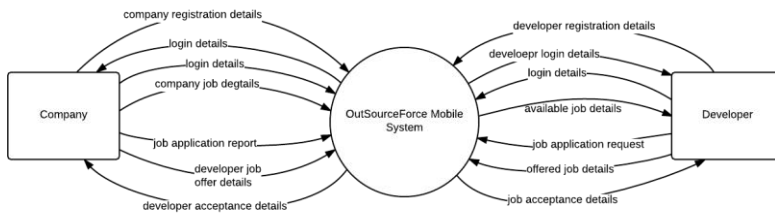
Once registered, developers can use the OutSourceForge app to login and browse existing jobs that match their skill set to see if there are any jobs that interest them. Once a developer finds a job that they are interested in, they can submit a job request to the system. The developer's details are then stored in the Job file alongside the job they have applied for. At the end of each day a Job Request report is generated for each company with jobs listed. This report displays the jobs that the company has listed, the developers that have submitted a job request and the developer details.

Companies can view the reports either online through a mobile device or using a web browser and browse the history of each developer that has applied for the job. They are then able to select a developer and send them an offer of employment through the app. The job is then updated to indicate that it is currently under offer so is not available to developers.

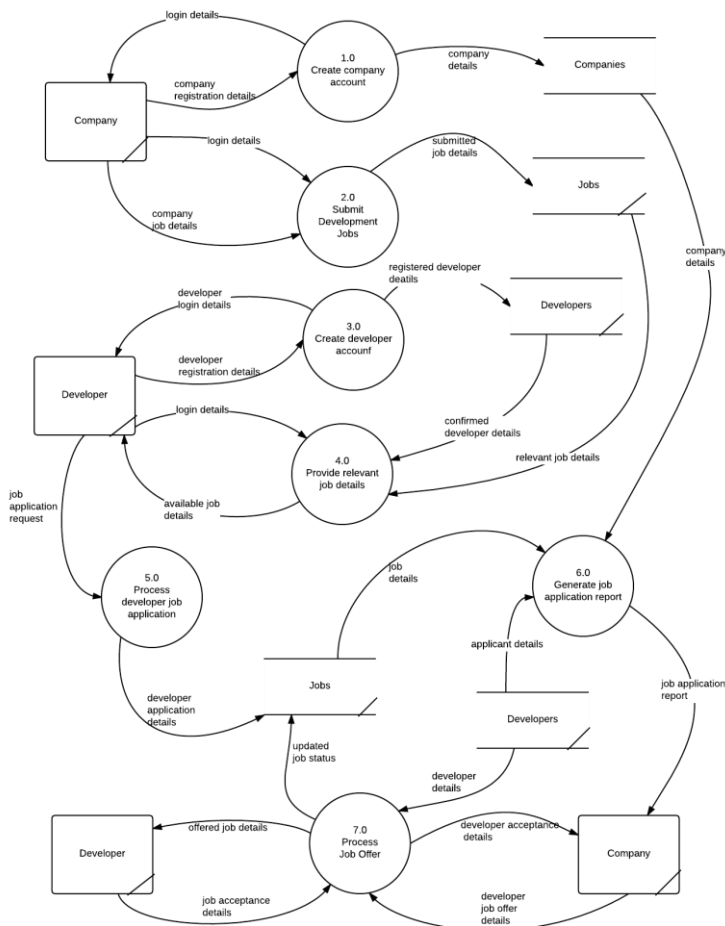
When an offer of employment is made, the developer is notified via an alert through the app. The developer then has 48 hours to either accept or decline the job. If the developer accepts the job, the job status is updated to 'taken' and the job is removed from the list of available jobs displayed by the app. If the developer declines the job or the offer times out, then the company is notified that the job has not been

taken and the job is updated to 'available' again. The developer account is also updated to indicate that they have either taken or declined a job.

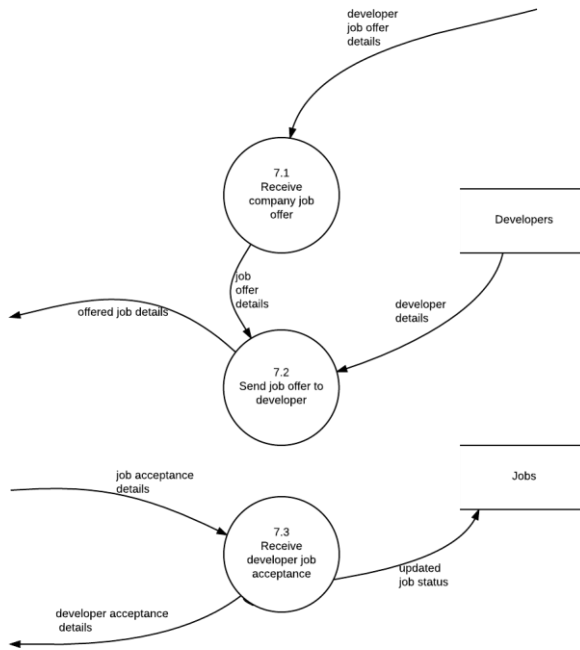
Context Diagram



Level 0 Data Flow Diagram



Level 1 Data Flow Diagram



Exercises

Question 1

JSV Banking is a regional bank that currently has branches throughout Western Australia. After several successful years of operating in regional WA, they would like to expand their operations and open offices throughout regional Australia. In order to make sure that they expand in a sustainable way and do not overstretch themselves, they would like to conduct a review of their current practices and find areas they can improve

As part of their strategy to increase their customer base, JSV Banking have decided to offer a special deal on home loans to new customers – current customers will not be eligible for this deal!

In order to get this deal, potential customers fill out an initial online application to make an appointment with a bank officer. This information will be used to check the customer database to verify that the applicant does not have any existing accounts with the bank. If the applicant does have an existing account an existing customer rejection notification will be sent, otherwise an appointment notification will be sent to the new customer.

The customer will then make an appointment with a bank officer who will complete a loan application with the customer. As part of this application, the bank officer will send the customer details to an external credit reporting agency to find out their current credit rating. If the customer credit rating is poor then the loan request is rejected and a poor credit rating notification is sent to the customer. If the credit rating is good

the customer is notified that they have preliminary approval for a loan so can put in an offer on the property they would like to purchase.

Once they put an offer on a property, the customer sends the bank officer the details of the property and the bank officer fills out a property valuation request form. This valuation request form is sent to an independent valuer who completes a valuation report and sends this back to the bank.

The valuation report is used to determine the final loan amount and a home loan contract is drawn up and sent to the customer. The valuation amount is recorded in the customer database. The valuation report is stapled to a copy of the contract and placed in the pending loan file.

If the customer is happy with the loan details, they sign the contract and send it back to the bank. The bank officer creates the new loan account and the actioned contract is placed into the current loan file. A letter outlining the loan account details is sent to the customer.

(a) Draw a Context Diagram for the system described above.

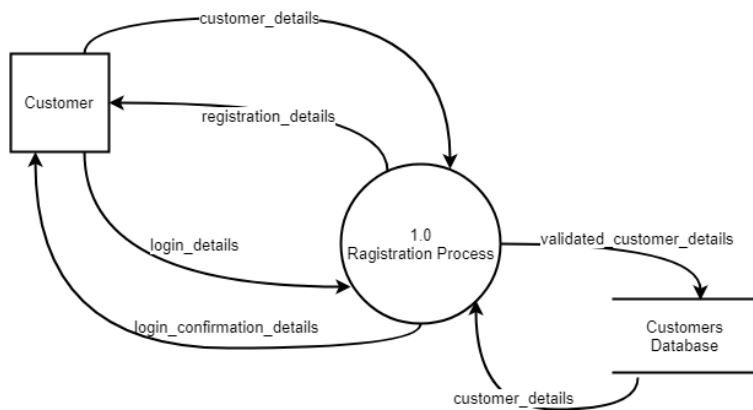
(b) Draw a Level 0 Data Flow Diagram (DFD) for the system described above.

Question 2

Vikki is developing an online ticketing system that will allow users to login and purchase tickets from their mobile devices.

During the registration process the user will enter their details into the system. These details will then be checked to ensure they are valid and, if valid, will be entered in the Customers Database. Registration details will then be sent to the user to allow them to login. When logging in, the username and password will be entered and checked against the Customers Database. If the details are correct, a response will be sent to the user confirming that login was successful.

The following partial level 0 Data Flow Diagram describes the customer registration process for Vikki’s app.



Level the Level 0 Data Flow Diagram above into a Level 1 Data Flow Diagram.

System Architecture

Fetch-Execute Cycle

- role of the following components of the central processing unit (CPU)
 - arithmetic logic unit (ALU)
 - control unit (CU)
 - registers
 - program counter
 - system clock
 - data, address and control bus
- purpose of the fetch-execute cycle
- stages of the fetch-execute cycle
 - fetch
 - decode
 - execute
 - store

Key points

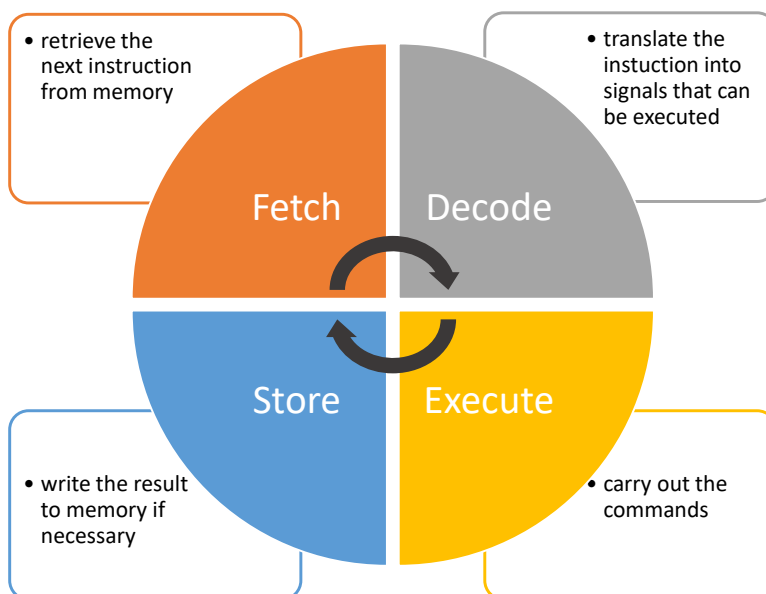
- Fetch-execute cycle describes the process that occurs when an instruction is executed by the CPU
- Need to know the role of each component in the CPU and what role each component plays within the fetch-execute cycle
- Useful simulation can be found at <https://peterhigginson.co.uk/LMC/>. See the Help section for more information on how it works.

Components of the CPU

- **Arithmetic Logic Unit (ALU)**: electronic circuit that performs basic arithmetic and logic operations. It receives two operands and an opcode that configures the ALU to perform the correct operation. The result of the operation is then placed in the accumulator (a specialised register) and the status flags are set.
- **Control Unit (CU)**: directs the operation of the processor by coordinating all parts of the processor to carry out the required instructions. The control unit interprets (or decodes) the current instruction (from the Instruction Register) and generates the necessary control signals to ensure the instruction is executed.
- **Registers**: small, high speed memory locations that temporarily hold data that is being used by the CPU. Some common registers include the program counter, address register, instruction register and the accumulator.
- **Program Counter (PC)**: a specialised register that stores the memory address of the **next** instruction to be executed. It is incremented after the next instruction has been fetched from memory.
NOTE: Some websites will use a different name for this register. Use the name Program Counter as that is what is in the syllabus.
- **System Clock**: a small crystal that vibrates at a specific speed that is used to control the timing of the operations of the CPU. The clock speed is measured in cycles per second (hertz)
- **Buses**: wires that are used to transfer data between components of the computer
 - Data bus: transfers actual data between components
 - Address bus: specifies the memory location to read from or write data to
 - Control bus: carries commands from the Control Unit to other components

Stages of the Fetch-Execute Cycle

- **Fetch:** the next instruction is retrieved from memory. The address currently stored in the Program Counter is loaded onto the Address Bus and the read wire is enabled to allow that location in memory to be read. The contents of that memory location are then placed on the Data Bus and transferred to the Instruction Register. The Program Counter is then incremented to point to the next memory location.
- **Decode:** the Control Unit reads the Instruction Register and interprets the instruction that needs to be carried out. This instruction is then placed on the Control Bus and passed to the required components (usually the ALU).
- **Execute:** the command is executed by the necessary components, such as the ALU.
- **Store:** the results of any calculations are placed in the accumulator, other registers or stored in RAM.



Virtualisation

- benefits of virtualisation
- types of platform virtualisation
 - desktop virtualisation
 - personal computer virtualisation
 - server virtualisation
 - storage virtualisation

Key Points

Virtualisation allows several self-contained "virtual" machines to run their own operating system and applications on a single physical "host" machine. Each virtual machine is self-contained and completely isolated from any other virtual machines, behaving as if it was a distinct physical computer with its own CPU, RAM, hard disk and network interface card (NIC). As far as the operating system, applications and other computers on the network are concerned, a virtual machine is the same as any other physical machine.

Commented [CA1]: Add image of server and PC virtualisation?

Type	Description	Benefits
Server virtualisation	multiple servers are hosted on a single, physical machine. Each server has its own operating system and is usually running a single application (e.g., a web server or mail server).	<ul style="list-style-type: none"> • Do not need to purchase as many expensive servers • Less maintenance required as fewer physical machines • Lower electricity costs • Less physical space required • Can vary processing power available as necessary
PC Virtualisation (Personal Computer)	a single physical computer runs multiple virtual PCs within the operating system of the host computer. Each virtual PC is self-contained and runs its own operating system and applications, separate from the other virtual PCs. This enables the user to switch between multiple OSs without the need to reboot their computer.	<ul style="list-style-type: none"> • Able to run multiple different configurations on a single PC • Able to run different operating systems on a single PC • Gives user ability to run software that only works on a different OS (e.g. run Windows software on a Mac)
Desktop Virtualisation	allows applications to be centralised at a data centre, where a server can host an entire desktop environment that each user can access using various levels of authentication.	<ul style="list-style-type: none"> • All files can be stored centrally, restricting user access to company data • Users can work from anywhere with an Internet connection • User environments can be centrally managed (i.e., OS and applications can be updated once in central location)
Storage virtualisation	allows for several physical storage devices to be combined so they appear to the user as a single logical storage device.	<ul style="list-style-type: none"> • Allows users to share storage locations • Allows company to have a variety of physical storage devices that appear to the users as a single location

Cloud Computing

- purpose of cloud computing
- advantages and disadvantages of cloud computing

Commented [CA2]: What is cloud computing and what is its purpose?

Service Models

There are three basic types of service models that are offered by cloud computing providers:

- **Infrastructure as a Service (IaaS)** - primarily encompasses the hardware and technology for computing power, storage, or other infrastructure delivered as an off-premises, on-demand service. The infrastructure for an organisation can be delivered on demand, and often includes costs such as licensing fees for operating systems and software installed on the computers.
- **Platform as a Service (PaaS)** - involves the delivery of a computing platform that can be used by developers to develop and run their own custom software. It typically involves an operating system, programming language, execution environment, database platform and web server.
- **Software as a Service (SaaS)** - comprises end-user applications that are delivered on-demand over the internet rather than as traditional, on-premises software. This type of software does not require any software to be installed on the computer - simply a browser and an internet connection. Some common SaaS applications include Gmail, Hotmail and Google Docs.

Deployment Models

There are three basic methods of delivering cloud services:

- **Private Cloud** - is cloud computing infrastructure that is hosted for a single organisation, with nobody else having access to those service.
- **Public Cloud** - infrastructure that is made available to the general public. Generally, they are either free or offered on a pay-per-use model.
- **Hybrid Cloud** - uses a combination of private and public cloud computing to suit the needs of an organisation.

Advantages/Disadvantages of Cloud Computing

Advantages	Disadvantages
<ul style="list-style-type: none"> • Do not need to pay for expensive infrastructure • Do not need to maintain physical infrastructure • Can access more processing power as needed • Greater reliability • Increased availability – can access network anywhere with Internet connection • Improved disaster recovery options 	<ul style="list-style-type: none"> • Do not have control over your data • Relying on the service provider to maintain infrastructure adequately • If service provider goes out of business, then may lose all data • If Internet connection fails, then cannot access data

Disaster Recovery

- purpose of disaster recovery plans
- types of disaster recovery tools, including:
 - online storage
 - incremental backup
 - full backup
 - RAID (Level 0, 1, 10)
 - uninterruptible power supply (UPS)

Key Points

- Disaster recovery plan is a written plan that describes the steps a company needs to take to restore operations as quickly as possible in the event of a disaster.
- Although a backup plan is important, by itself it does not constitute a disaster recovery plan – you still need to be able to restore the data from any backups that you have
- Need to practice your disaster recovery plan

Disaster Recovery Tools

Backups

- Involves making a copy of your data on a separate device so that it can be recovered in case of disaster.
- Backup should be stored in a different physical location, preferably offsite
- **Online storage:** use of an external provider (and cloud computing) to store data. The external provider is then responsible for maintaining backups of the data (e.g., OneDrive, Google Drive)
- **Full backup:** a copy of all the files on the system is made. This can take a long time if there is a large amount of data to backup, but is much easier to restore as you only need a single backup to restore all data
- **Incremental backup:** only files that have been changed since the last backup are copied. This provides an historical record of changes to files and takes less time than a full backup. Restoring data is more difficult as it requires having all the previous backups to make a full recovery.

RAID (Redundant Array of Inexpensive Devices)

- Uses a combination of data striping and data mirroring to store data across several physical disks
- Can give performance and reliability improvements
- **RAID 0:** Uses *data striping* to store data across multiple disks to improve performance. Data is split into chunks, and each chunk is then stored on a different device. As data is stored on multiple devices, multiple parts of each file can be read/written at the same time, improving performance. RAID 0 does not provide any redundancy, however, so if one disk fails then all data is lost.
- **RAID 1:** Uses *data mirroring* to allow data to be duplicated across multiple disks. Data is written to both disks at the same time, thus providing redundancy in case one disk fails. Reading performance can be improved for multiple users as users are able to read from either disk.
- **RAID 10:** Uses a combination of *data striping* (RAID 0) and *data mirroring* (RAID 1) to provide both improved performance and greater redundancy. RAID 10 requires a minimum of 4 disks.

Uninterruptible Power Supply

- Power supply that will continue to provide power in case of a disaster that causes a power outage.
- Designed to provide enough emergency power to shut down equipment (e.g. servers) properly to prevent data loss.
- A UPS is **not** designed to allow the business to keep operating until power is restored.

Managing Data

Database Concepts

- database management system concepts, including:
 - data definition
 - data duplication
 - data integrity, including: referential integrity, domain integrity and entity integrity
 - data redundancy
 - data anomalies, including: insert, delete and update
 - data manipulation
 - data security
- create a working relational multi-table database using:
 - data types
 - relations
 - primary, composite and foreign keys
 - referential integrity
 - relationships, including: set cascade inserts, updates and deletes
 - cardinality (1:1, 1:M, M:1, M:N)
 - validation rules
 - forms
 - reports

Data Integrity:

- Data integrity refers to the validity and accuracy of the data that is stored in a database.
- There is a difference between validity and accuracy. Validity checks to see that the data entered meets the business rules (e.g., a birthdate cannot be in the future). Accuracy means that the data is correct (e.g., the correct birthdate has been entered).
- A DBMS can only check the validity of the data – it has no way of being able to check the accuracy of data that has been entered.
- **Referential integrity:** Referential integrity refers to the relationships between tables in a database. This ensures that each foreign key refers to a valid primary key in the related table (or is null if there is no relationship between the records)
- **Domain integrity:** Domain integrity specifies the set of values that are valid for a column and determines whether null values are allowed. Domain integrity can be enforced by validity checking and by restricting the data type, format and/or range of possible values in a column.
- **Entity integrity:** Entity integrity refers to the records within a table. This ensures that every table has a primary key that is unique to each record and is not null.

Exercise

Table: Cabin				
cabin_id	name	beds	bathrooms	pets
1	Eagle	4	1	Yes
2	Wren	4	1	No
3	Parrot	8	2	No

Table: Customer			
customer_id	first_name	last_name	email
1001	Jenny	Lane	jenny88@hmail.com
1027	Max	Peterson	peterston@vipee.com
1384	Allan	Fowler	afowler@hmail.com.au

Table: Booking				
booking_id	date_in	date_out	customer_id	cabin_id
56852	11/10/2020	14/10/2020	1001	1
57823	15/12/2020	26/12/2020	1384	3
69825	16/12/2020	20/12/2020	1001	1

Using examples from the tables above, describe the following data integrity terms.

(a) Referential integrity:

(b) Domain integrity:

(c) Entity Integrity:

Normalisation

- Normalisation is the process of identifying and eliminating data anomalies and redundancies, thereby improving data integrity and efficiency for storage in a relational database. This process is designed to remove repeated data and improve database design.

Data Anomalies

- Consider the data in the table below. This unnormalised data can cause problems when data is updated, added or deleted.

Licence	FirstName	LastName	Email	Registration	Make	ManufacturerWeb
19289385	John	Smith	jsmith@combi.net	1COB 293	Ford	www.ford.com.au
19289385	John	Smith	jsmith@combi.net	1QAZ 889	Toyota	www.toyota.com.au
19289385	John	Smith	jsmith@combi.net	1CCT 441	Mazda	www.mazda.com.au
26453791	May	Hogarth	mhogarth@combi.net	1COB 293	Ford	www.ford.com.au
28852462	Peter	Jones	pjones@combi.net	1WRC 634	Toyota	www.toyota.com.au

Update anomaly

- An update anomaly occurs when you try to update data that is stored in multiple locations.
- If all records are not updated, then data becomes inconsistent and/or inaccurate
- For example, if John Smith updates his email address then **all 3** occurrences need to be updated

Delete anomaly

- Occurs when by deleting one piece of data you delete the only instance of another piece of data
- For example, if the Peter Jones is removed from the database, the details of the car 1WRC 634 will be lost

Insert anomaly

- Occurs when data cannot be added because only part of the data is available
- For example, if a new car is added, but no driver allocated then we would be unable to add the car as we need all the information to create a new record

Normalisation to 3rd Normal Form

- Steps to normalisation of data:
 1. Ensure data is in the form of a relation
 2. Convert data to 1st Normal Form
 3. Convert data to 2nd Normal Form
 4. Convert data to 3rd Normal Form

Convert data to a Relation

- In order for data to be in the form of a relation it must:
 1. Have no repeated attributes
 2. All cells must be atomic (that is, they must only contain a single piece of data)

Repeated Fields

Licence	FirstName	LastName	Registration 1	Registration 2	Registration 3
19289385	John	Smith	1COB 293	1QAZ 889	1CCT 441
26453791	May	Hogarth	1COB 293		

The above table is **not** in the form of a relation as it has repeating fields. The Registration field is repeated multiple times.

Non-atomic Field

Licence	FirstName	LastName	Registration
19289385	John	Smith	1COB 293, 1QAZ 889, 1CCT 441
26453791	May	Hogarth	1COB 293

The above table is **not** in the form of a relation as one of the fields is not atomic. The Registration field for John Smith has information about three different cars.

Relation

Licence	FirstName	LastName	Email	Registration
19289385	John	Smith	jsmith@combi.net	1COB 293
19289385	John	Smith	jsmith@combi.net	1QAZ 889
19289385	John	Smith	jsmith@combi.net	1CCT 441
26453791	May	Hogarth	mhogarth@combi.net	1COB 293

The above table **is** in the form of a relation as all fields are atomic and there are no repeating fields.

NOTE: This data is not normalised and would not make a good database structure, but we can now start the process of normalisation.

1st Normal Form

- To be in 1st Normal Form, we must:
 - ensure that all fields are atomic
 - remove all repeating attributes
- Each relation that is formed will have a primary key
- The relation formed from the non-repeating attributes will have a foreign key to the relation formed from the repeating attributes
- The primary key for the relation for the non-repeating fields will now be a composite key comprising the primary key from the non-repeating relation and the repeating relation

2nd Normal Form

- To be in 2nd Normal Form, we must:
 - Be in 1NF
 - Have no partial dependencies
- Partial dependencies occur when a non-key attribute is only dependent on part of the composite key, rather than on the entire composite key
- If a relation does not have a composite key (that is the primary key is made up of a single attribute) then it must already be in 2NF.

3rd Normal Form

- To be in 3rd Normal Form, we must:
 - Be in 2NF
 - Have no transitive dependencies
- All non-key fields in a relation must be fully functionally dependent on the primary key
- Transitive dependencies occur when a non-key field is dependent on a field other than the primary key

Normalisation Example

Relation

- Consider the following data. Is it in the form of a relation?

Student Num	First Name	Last Name	Course	Course Name	Result	Result Description
10010504	David	Brown	MATH1001	Mathematics 1A	A	Highly Skilled
10010504	David	Brown	MATH1002	Mathematics 1B	B	Skilled
10010504	David	Brown	COMP1001	Computing 1A	A	Highly Skilled
10020423	James	Stanton	MATH1001	Mathematics 1A	C	Competent
10020423	James	Stanton	COMP1001	Computing 1A	C	Competent
23521461	Debbie	Tainton	MATH1001	Mathematics 1A	B	Skilled
23521461	Debbie	Tainton	MATH1002	Mathematics 1B	A	Excellent
23521461	Debbie	Tainton	COMP1001	Computing 1A	A	Excellent
24352494	Alison	Brown	MATH1002	Mathematics 1B	C	Competent
24352494	Alison	Brown	COMP1001	Computing 1A	A	Excellent

- We can write this as a relation in the form:
StudentResults(StudentNum, FirstName, LastName, Course, CourseName, Results, ResultDescription)

Convert to 1NF

- Firstly, we need to check that all attributes are atomic.
- Then remove all repeating attributes and place them in another relation.

Student Num	First Name	Last Name
10010504	David	Brown
10020423	James	Stanton
23521461	Debbie	Tainton
24352494	Alison	Brown

StudentNum	Course	Course Name	Result	Result Description
10010504	MATH1001	Mathematics 1A	A	Highly Skilled
10010504	MATH1002	Mathematics 1B	B	Skilled
10010504	COMP1001	Computing 1A	A	Highly Skilled
10020423	MATH1001	Mathematics 1A	C	Competent
10020423	COMP1001	Computing 1A	C	Competent
23521461	MATH1001	Mathematics 1A	B	Skilled
23521461	MATH1002	Mathematics 1B	A	Excellent
23521461	COMP1001	Computing 1A	A	Excellent
24352494	MATH1002	Mathematics 1B	C	Competent
24352494	COMP1001	Computing 1A	A	Excellent

- We can also write this in the form of relation definitions:
Student(StudentNum, FirstName, LastName)
StudentCourse(StudentNum FK, Course FK, CourseName, Result, ResultDescription)

Convert to 2NF

- Check for and remove any partial dependencies.
- Partial dependencies will only occur in a relation that has a composite key, so Student is already in 2NF.

Student Num	First Name	Last Name
10010504	David	Brown
10020423	James	Stanton
23521461	Debbie	Tainton
24352494	Alison	Brown

Course	CourseName
MATH1001	Mathematics 1A
MATH1002	Mathematics 1B
COMP1001	Computing 1A

StudentNum	Course	Result	Result Description
10010504	MATH1001	A	Highly Skilled
10010504	MATH1002	B	Skilled
10010504	COMP1001	A	Highly Skilled
10020423	MATH1001	C	Competent
10020423	COMP1001	C	Competent
23521461	MATH1001	B	Skilled
23521461	MATH1002	A	Excellent
23521461	COMP1001	A	Excellent
24352494	MATH1002	C	Competent
24352494	COMP1001	A	Excellent

- We can also write this in the form of relation definitions:
Student(StudentNum, FirstName, LastName)
Course(Course, CourseName)
StudentCourse(StudentNum FK, Course FK, Result, ResultDescription)

Convert to 3NF

- Finally, we need to check there are no transitive dependencies.
- In this case, the result description is dependent on the result, not the course.

Student Num	First Name	Last Name
10010504	David	Brown
10020423	James	Stanton
23521461	Debbie	Tainton
24352494	Alison	Brown

Course	CourseName
MATH1001	Mathematics 1A
MATH1002	Mathematics 1B
COMP1001	Computing 1A

StudentNum	Course	Result
10010504	MATH1001	A
10010504	MATH1002	B
10010504	COMP1001	A
10020423	MATH1001	C
10020423	COMP1001	C
23521461	MATH1001	B
23521461	MATH1002	A
23521461	COMP1001	A
24352494	MATH1002	C
24352494	COMP1001	A

Result	Result Description
A	Highly Skilled
B	Skilled
C	Competent

- We can also write this in the form of relation definitions:
Student(StudentNum, FirstName, LastName)
Course(Course, CourseName)
StudentCourse(StudentNum FK, Course FK, Result FK)
Result(Result, ResultDescription)

Exercises

Question 1

Mt Barker High School has recorded the results from its annual swimming carnival in an Excel spreadsheet. An extract of that spreadsheet is below. (NOTE: Each student can only be a member of one team and can only compete in one age group).

Event	Age	Gender	Distance	Stroke	Competitor	Team	Time	Place	Points
1	U/13	M	50m	Freestyle	David Leblanc	Dolphins	00:32.9	1	16
1	U/13	M	50m	Freestyle	Barry Hu	Eels	00:36.3	2	13
1	U/13	M	50m	Freestyle	Thomas Black	Sharks	00:40.6	3	11
35	U/17	M	100m	Breaststroke	Jacob Morris	Eels	01:24.7	5	8
35	U/17	M	100m	Breaststroke	Chase Sanders	Dolphins	01:33.1	6	7
35	U/17	M	100m	Breaststroke	Oliver Hahn	Sharks	01:41.0	7	6
35	U/17	M	100m	Breaststroke	Simon Wood	Rays	01:51.2	8	5
55	U/21	M	100m	Breaststroke	Brett Hobbs	Eels	01:15.6	1	16
55	U/21	M	100m	Breaststroke	Eric Lee	Rays	01:18.5	2	13
55	U/21	M	100m	Breaststroke	Barry Stokes	Sharks	01:32.9	5	8
63	U/21	M	100m	Backstroke	Brett Hobbs	Eels	01:12.5	2	13
63	U/21	M	100m	Backstroke	Brendan Serrano	Dolphins	01:13.2	3	11
63	U/21	M	100m	Backstroke	Barry Stokes	Sharks	01:18.1	6	7
63	U/21	M	100m	Backstroke	Eric Lee	Rays	01:21.4	7	6

Normalise the data to third normal form (3NF). Give your answer in the form of relation definitions.

Question 2

The following excerpt shows data that is stored by a bank manager about the customer accounts that he oversees.

Customer ID	Customer Name	Account Number	Account Type	Product Name	Current Balance	Interest Rate	Date Opened	Opened By	Branch
10191	David Burgess	4485 6737 9486 1104	Credit card	Blue VISA	\$50,470.96	21.40%	Dec 23, 2016	Finn Compton	Broome
10191	David Burgess	51931146	Home Loan	Investment One	\$25,611.30	5.24%	Dec 20, 2016	Margaret Wilkins	Broome
10191	David Burgess	79573746	Savings	Everyday Savings	\$88,647.49	0.01%	Apr 20, 2018	Margaret Wilkins	Broome
10191	David Burgess	41425482	Home Loan	First Homebuyer	\$26,173.15	4.03%	Aug 26, 2017	Margaret Wilkins	Broome
10228	Alice Green	89818296	Savings	Saver Plus	\$85,739.02	0.50%	Dec 16, 2017	Adele Perry	Dunsborough
10228	Alice Green	54456999	Savings	Everyday Savings	\$25,060.73	0.01%	Jan 26, 2018	Adele Perry	Dunsborough
10228	Alice Green	73663187	Home Loan	Investment One	\$47,096.15	5.24%	Dec 16, 2016	Adele Perry	Dunsborough
10393	Stephanie Cunningham	5211 7675 1802 1926	Credit card	Platinum MasterCard	\$69,794.45	21.60%	Jul 8, 2017	Hadley Nguyen	Karratha
10393	Stephanie Cunningham	66678786	Savings	Saver Plus	\$87,059.64	0.50%	May 28, 2017	Hadley Nguyen	Karratha
10393	Stephanie Cunningham	52787122	Home Loan	Investment One	\$1,288.30	5.24%	May 12, 2018	Margaret Wilkins	Broome
10409	Mercedes Hendricks	61193735	Savings	Saver Plus	\$22,718.79	0.50%	Feb 9, 2017	Finn Compton	Broome
10589	Emily Goodwin	81269568	Home Loan	First Homebuyer	\$3,904.93	4.03%	May 7, 2017	Margaret Wilkins	Bunbury
10591	Wayne Carter	84313126	Home Loan	Investment One	\$88,073.16	5.24%	Dec 14, 2017	Margaret Wilkins	Bunbury
11011	Neville Moran	4929 4031 8905 9823	Credit card	Blue VISA	\$71,916.91	21.40%	Dec 25, 2017	Adele Perry	Dunsborough

(a) Insert anomaly

(b) Delete anomaly

(c) Update anomaly



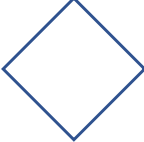
(d) Normalise the data to 3rd Normal Form (3NF)

Entity Relationship Diagrams

- data modelling using Chen’s notation entity relationship (ER) diagrams
- analyse existing ER diagrams
- create accurate ER diagrams
- create a model of a database solution using Chen’s notation entity relationship (ER) diagrams
- resolve complex many to many (M:N) relationships in a multi-table relational database system (three or more entities)

- An Entity Relationship Diagram (ERD) provides a graphical representation of the interrelationships between entities in a database.
- Provides an easy, graphical way to visualise how a database is structured and how each entity is related to others.
- There are many variations on how to draw ERDs, but the WACE course uses Chen notation. **You must use this in your exam, or you will lose marks.**

Symbols

Symbol	Name	Description
	Entity	Indicate objects, people or things about which data is kept: <ul style="list-style-type: none"> • Must be labelled using a singular noun
	Attribute	Represent the properties of each entity and describe the data that is being stored about an entity. <ul style="list-style-type: none"> • Identify the Primary Key (or Composite Key) by underlining the attribute name • Identify a Foreign Key by adding the letters FK
	Relationship	Describe the links between entities. It is also used to indicate the cardinality or type of relationship. <ul style="list-style-type: none"> • Should be labelled using a verb that describes the relationship • Cardinality can be: <ul style="list-style-type: none"> ○ 1:1 – one to one ○ 1:M – one to many ○ M:1 – many to one ○ M:N – many to many (also written as M:M) NOTE: All many to many relationships should be resolved (see next section)

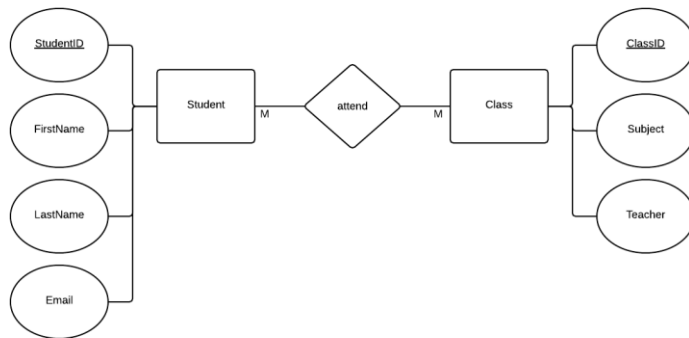
Creating an ERD

To create an ERD:

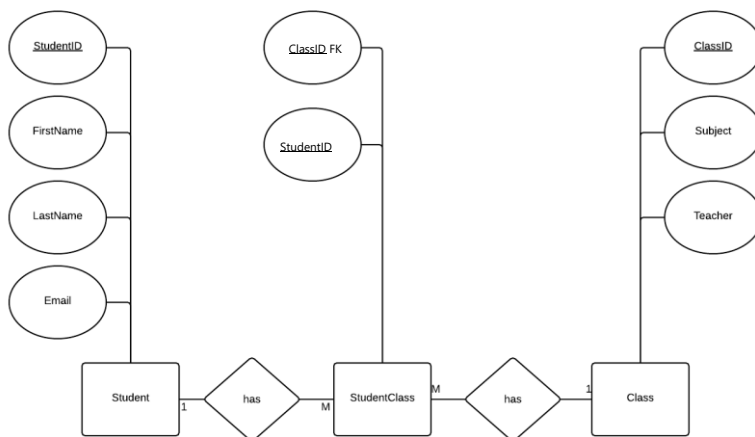
1. Identify the entities that need to have information stored about them
2. Identify the relationships between the entities
3. Identify the cardinality of the relationships
4. Create a primary key for each entity
5. Create any necessary foreign keys
6. Determine any other non-key attributes for each entity

Many-to-many relationships

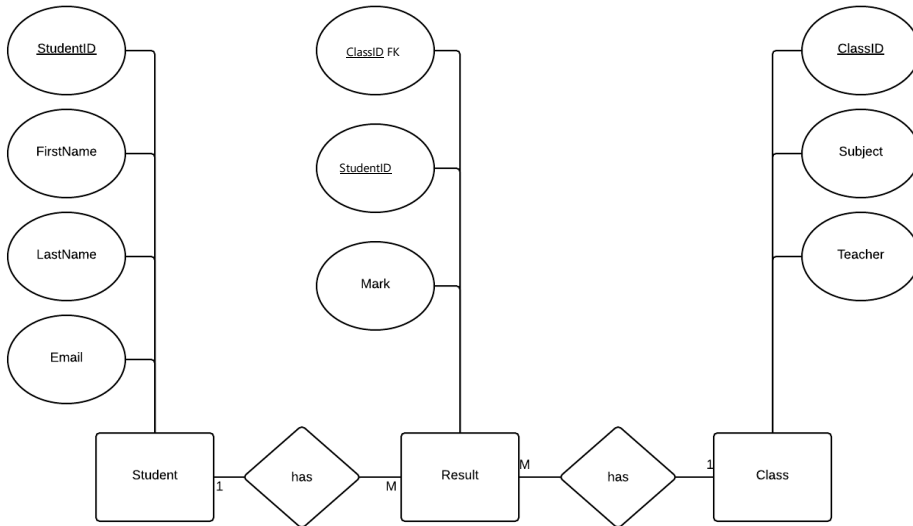
- When creating a relational database, it is impossible to directly implement a many-to-many relationship. Instead, we need to *resolve* any many-to-many relationships we have in our ERD.
- To resolve the many-to-many relationship, we need to place a linking entity between the two existing entities that will allow us to create two one-to-many (1:M) relationships.
- Consider the following M:N relationship



Each student can attend many classes, and each class has many students. To convert this relationship into a form that can be implemented in a relational database, we need to insert an entity to join the two existing entities. This joining (or bridging) entity will have a foreign key to both the Student entity and the Class entity. We could then either use both these foreign keys to create a composite key or add another unique identifier to use as a primary key. The resulting relationships are shown in the ER diagram below.



Once we have our bridging entity, we may want to store more information, or have a more meaningful name that we could use. For example, each student will have a mark for each subject, so we could call the bridging entity Result and record the mark for each subject in that entity.

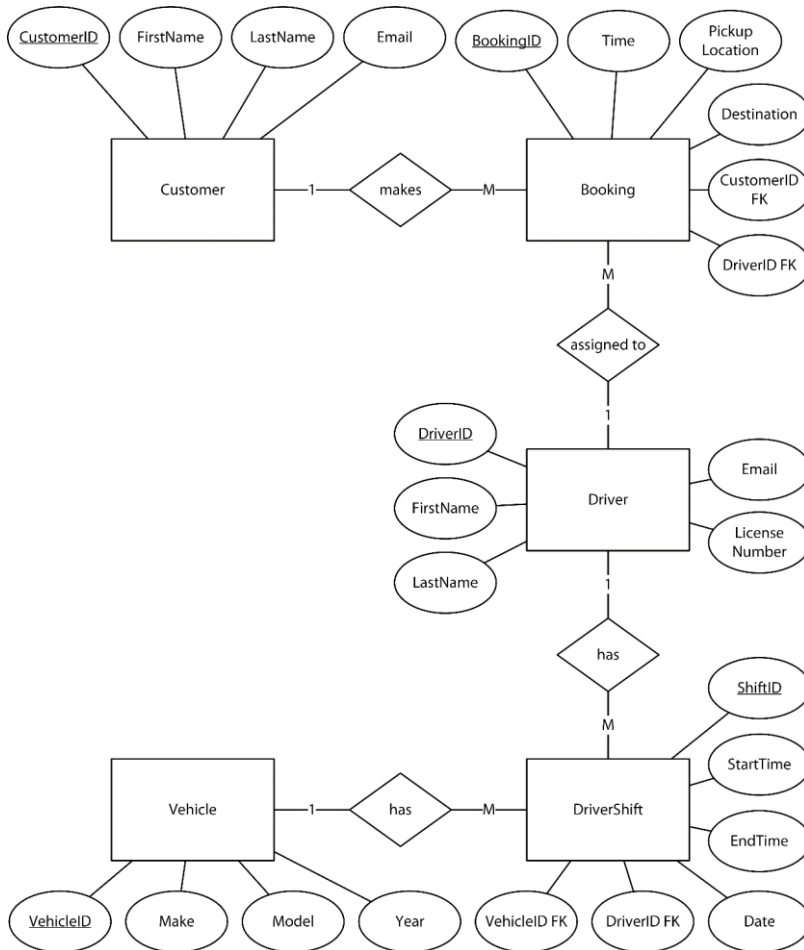


Sample ERD

Jake is starting a new taxi service and wants to create a database to store information about all his customers, their bookings and his drivers. He has provided you with the following information:

- Customers will need to register with an app on their phone, entering their first name, last name and email address.
- Jake has a fleet of 8 cars that are shared by the drivers
- When a driver starts a shift, they are assigned one of the available cars. This may change each shift, and Jake needs to keep a record of which car each driver was using for each past shift.
- When a customer makes a booking, they enter the booking details (time and location of pickup, the destination and number of passengers). The booking system will then assign them the first driver who is free at that time.

Draw an entity relationship diagram for Jakes Taxi Service. Include all necessary attributes and resolve any many to many relationships.



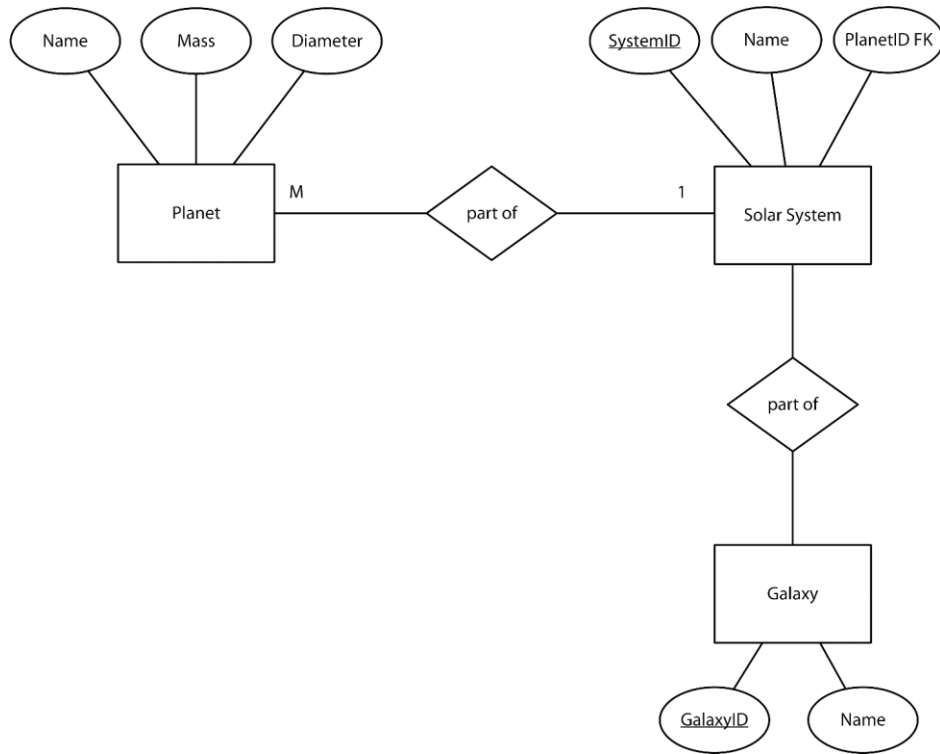
Exercises

Question 1

Consider the following relationships.

- A solar system is made up of many planets.
- Many solar systems make up a galaxy.

Identify and describe four errors in the Entity-Relationship Diagram (ERD) below.



i _____

ii _____

iii _____

iv _____

Question 2

JSV Banking has a dedicated IT department based at their head office in Bunbury and would like a database to keep track of all the projects that each employee is working on. The IT department is made up of several different sections (for example, Web Development) and each section will have multiple projects running at any one time. For most projects, more than one employee will be needed, and each employee will need to divide their time across several different projects. Each employee will also work in a specific section of IT, however, may be assigned to projects across several different sections as needed.

Draw an Entity Relationship Diagram (ERD) to model the required database. You should include all relationships, primary keys and foreign keys, attributes and resolve any many to many relationships. (15 marks)

Question 3

Jake has started a Home Maintenance business that provides maintenance and gardening services to people throughout Perth. When a customer requests a job, he looks through his list of sub-contractors and allocates the job to the most suitable person.

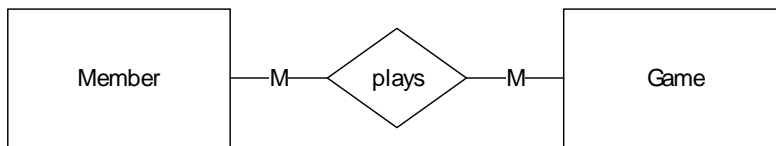
When the job is completed, the customer is issued with an invoice (such as the one shown in the image below). Currently he uses an Excel spreadsheet to create each invoice, but as his business has grown, he has found it increasingly time consuming to create each invoice. To solve this problem, Jake has decided to upgrade his invoicing system to use a database to store all the necessary data.

Jake's Home Maintenance and Garden Services		INVOICE	
ABN:5684 599 214		Invoice Num	INV0018656
		Date	15/07/2020
		Customer ID	CUS1985768
Customer:	Peter Jones		
Address:	14 Bell Rd Claremont, WA, 6010		
Phone:	0458 684 286		
Email:	pjones@eenet.net.au		
Contractor:	John Smith		
Qty	Description	Unit Price	Line Price
100	Pickets for fence	\$5.75	\$575.00
2	White paint	\$58.00	\$116.00
1	Wood treatment	\$75.00	\$75.00
8	Labour	\$80.00	\$640.00
		Subtotal	\$1,406.00
		GST	\$140.60
		Total	\$1,546.60
Payment must be made within 7 days of the invoice date.			

Using the information on the previous page, draw an Entity-Relationship Diagram (ERD) to show how his new database will be structured. Resolve any many-to-many relationships and include all necessary attributes.

Question 4

The local computer gaming club wants to start a database of all the games that its members have participated in. They want to keep track of the member's details, the details of each game, and the date, time and score for each game that the member played. They have provided you with this partial Entity Relationship (ER) Diagram. Resolve the diagram, including any primary and foreign keys that may be necessary.



Legal/Ethical Issues

- role of law and ethics in the storage and disposal of personal data, including: the impact of privacy laws in Australia on the storage and distribution of data
-
- When storing personal data, developers need to be aware of both their legal and ethical obligations.
 - Legal obligations are covered by the Australian Privacy Act 1988, and specifically by the Australian Privacy Principles. (see <https://www.oaic.gov.au/privacy/australian-privacy-principles/> for more information)
 - Some legal obligations include:
 - Must store personal data securely
 - Must allow people access to data stored about them
 - Must notify people as to what data is being stored about them
 - Personal information that is no longer needed must be disposed of in a secure, timely and reasonable manner
 - Data must only be used for purposes that the user has agreed to
 - Ethical issues may include:
 - Ensuring data is used appropriately
 - Personal information is not used to negatively impact on an individual

Exercise

Question 1

Peter owns a second-hand car dealership with a yearly turnover of approximately \$3 million dollars. He collects, stores and disposes of car registration and drivers licence data as a normal part of selling cars. The registration and licence data that he collects is saved to a flash drive so he can easily exchange it with an insurance dealer he knows so that his friend can sell insurance to people who have just bought a car.

(a) Explain two ways that Peter may be breaking the law.

i _____

ii _____

(b) Explain one ethical issue with what Peter is doing.

Question 2

AussieToys is an online toy store that sells Australian made toys across the world. As part of their sales process, they collect information about each of their customers so they can personalise the shopping experience for each customer.

(a) Identify the government act that regulates the use of personal information.

(b) Describe two aspects of this act that AussieToys needs to consider when collecting personal information about their customers.

i

ii

Programming

Key Concepts

- characteristics of simple data types:
 - integer
 - real (floating point number)
 - Boolean
 - character
- characteristics of complex data types:
 - string
 - one-dimensional arrays
 - records
- programming concepts, including:
 - constants
 - variables (local, global, parameters)
 - appropriate naming conventions for variables
 - control structures
 - stubs
 - statements
 - modularisation
 - functions
 - scope and lifetime of identifiers, including: parameter passing (value and reference)
- types of program or code errors, including:
 - syntax errors
 - run-time errors
 - logical errors

Data Types

- **Simple** data types:
 - **Integer**: a whole number. The number of bytes used to represent an integer determines the size of the integers that can be stored.
 - **Real** (floating point number): number that includes a decimal place
 - **Boolean**: used to represent data that is either *True* or *False*
 - **Character**: a single unit of data used to store a single alphanumeric entry, such as a letter, digit or punctuation. In ASCII, this is a single byte.
- **Complex** data types:
 - **String**: a collection of characters
 - **Array**: stores multiple values of the same data type
 - **Record**: stores multiple different pieces of data (possibly of different types) about a single item

Programming Concepts

- **Constants:** named memory location whose literal value does not change throughout the execution of a program
- **Variables:** named memory location whose contents can change and be used during the execution of a program
- **Scope of Identifiers:** the scope of an identifier indicates in which sections of the code an identifier is accessible
 - **Local:** are only accessible within the section of code that they have been declared
 - **Global:** are accessible throughout the entire program
- **Modularisation:** the process of breaking a program down into several smaller sections that can be used to perform a specific purpose
- **Parameters:** an argument that can be passed into a module
 - **Value parameters:** a copy of the actual data is passed to the module that is being called. Any changes to the parameter inside the module do not affect the original value
 - **Reference parameters:** a pointer to the variable's memory location is passed to the module being called. Any changes to the parameter cause the original value to be changed
- **Types of programming errors:**
 - **Syntax errors:** an error that is caused by program statements that do not conform to the rules of the language
 - **Run-time errors:** an error that occurs during the execution of a program that prevents the program from running correctly.
 - **Logic errors:** an error that occurs when the program executes however produces unexpected output. Logic errors are often the most difficult to debug and software requires substantial testing to ensure there are no logic errors.

Pseudocode

- use pseudocode to represent a programming solution
 - apply, using pseudocode and a programming language, the following programming concepts:
 - constants
 - variables (local, global, parameters)
 - naming conventions for variables
 - stubs
 - statements
 - modularisation
 - functions
 - parameter passing (value and reference)
 - one-dimensional arrays
 - apply, using pseudocode and a programming language, the following control structures:
 - sequence
 - selection
 - one-way (if then)
 - two-way (if then else)
 - multi-way (case, nested if)
 - iteration
 - test first (while)
 - test last (repeat until)
 - fixed (for)
-
- Pseudocode uses a form of structured English to describe an algorithm
 - Some tips for writing good pseudocode:
 - Make sure your algorithm is language independent. Good pseudocode should be able to be translated into any suitable programming language
 - Write only one statement per line
 - Use capital letters for keywords
 - Make sure you indent your code to show the structure of the code
 - Avoid making your code unnecessarily complex
 - When naming variables:
 - Begin the variable name with lower case letters
 - Should always begin with a letter
 - Names should only contain letters or numbers
 - Use a capital letter to begin new word (Camel Case)
 - Do not add spaces to a name
 - Use the symbol ← to indicate an assignment statement
 - Use the symbol = to indicate a comparison statement
 - **NOTE:** There is no set standard for exactly how to write pseudocode, but you need to make sure it is clear and easy for the marker to understand. The sections below show the recommended way of writing pseudocode.

Common Commands

User input	INPUT(num) READ(num)
User output	OUTPUT("Hello world!") PRINT("Hello world!")
Assignment	←
Equals (comparison)	=
Not equal to	<> or !=
Greater than	>
Greater than or equal to	>=
Less than	<
Less than or equal to	<=
Modulus (remainder)	MOD
OR	x < 1 OR x > 10
AND	x > 1 AND x < 10
Arrays	scores ← [] scores[0] ← 15 scores[1] ← 16 scores.append(12) scores.push(12) scores.length

Control Structures

- Control structures are used to control the execution flow of a program, allowing a program to branch or repeat lines of code.

Selection

One-way selection	If a condition is TRUE, run some code, otherwise no code is run, and the statement terminates	<pre> speed ← 50 IF speed > 60 THEN PRINT("You are speeding") END IF </pre>
Two-way selection	If a condition is TRUE, run some code, otherwise if the condition is FALSE run some other code.	<pre> speed ← 50 IF speed > 60 THEN PRINT("You are speeding") ELSE PRINT("You are not speeding") END IF </pre>
Multi-way selection	Test a condition against a variety of different outcomes and perform an action based on which outcome is TRUE	<p>Method 1 – IF...ELSE IF...ELSE</p> <pre> speed ← 50 IF speed < 20 THEN PRINT("You are going too slow") ELSE IF speed > 60 THEN PRINT("You are speeding") ELSE PRINT("You are not speeding") END IF </pre> <p>Method 2 – CASE statement</p> <pre> speed ← 50 CASE speed OF < 20: PRINT("You are going too slow") <= 60: PRINT("You are not speeding") > 60: PRINT("You are speeding") END CASE </pre>

Iteration (or Repetition)

Test-first loop (WHILE)	A series of instructions are executed while the test condition is TRUE. The condition is tested before the instructions are executed.	<pre> num ← 0 WHILE num < 10 PRINT("The number is " + num) num ← num + 1 END WHILE </pre>
Test-last loop (REPEAT UNTIL)	A series of instructions are executed until the test condition is true (that is, it continues executing while the condition is FALSE). The condition is tested at the end of each loop.	<pre> num ← 0 REPEAT PRINT("The number is " + num) num ← num + 1 UNTIL num = 10 </pre>
Fixed loop (FOR)	Instructions are repeated a fixed number of times. These loops are used when the number of required iterations is known.	<pre> FOR num ← 1 TO 10 PRINT("The number is " + num) END FOR </pre>

Modularisation

- Modularisation is a methodology that involves breaking a problem down into smaller, less complex parts.
- Benefits of modularisation include:
 - Allows code to be reused, thereby reducing code repetition
 - Allows more people to work on a project – each person can work on separate modules
 - Breaking a large, complex problem down into smaller problems makes it easier to solve
 - Makes it easier to read algorithms and programs
 - Makes it quicker and easier to find errors

Example

Without Modularisation	With Modularisation
<pre> MODULE Main INPUT(length) INPUT(height) area1 ← length * height INPUT(length) INPUT(height) area2 ← length * height INPUT(length) INPUT(height) area3 ← length * height total ← area1 + area2 + area3 OUTPUT("The total area is", total) END Main </pre>	<pre> MODULE Main area1 ← 0 area2 ← 0 area3 ← 0 Call CalculateArea(area1) Call CalculateArea(area1) Call CalculateArea(area3) total ← area1 + area2 + area3 OUTPUT("The total area is", total) END Main MODULE CalculateArea(area) INPUT(length) INPUT(height) area ← length * height END CalculateArea </pre>

- The code on the left repeats the same lines of code three times where it calculates the area based on the length and height.
- The code on the right reduces this repetition by moving those lines of code to a separate module.

Parameter Passing

- We use **parameters** to pass values between modules
- There are two types of parameters:
 - **Value parameters:** a copy of the actual data is passed to the module that is being called. Any changes to the parameter inside the module do not affect the original value
 - **Reference parameters:** a pointer to the variable’s memory location is passed to the module being called. Any changes to the parameter cause the original value to be changed
- Generally, a value parameter is used to pass values into a module, and a reference parameter is used to pass a value from a module back to the calling module.
- In most programming languages, simple data types (including strings) will be passed by value and complex data types (such as arrays and records) will be passed by reference.

Modules vs Functions

- When breaking code up into sub-routines, we can create either a module or a function
- A **module**:
 - Should be given a meaningful name that describes its purpose
 - Can return none, one or many values through reference parameters
 - Must be called using the keyword CALL
 - Can have INPUT and OUTPUT statements
- A **function** is a module that returns a single value through the function name
 - Should be given a meaningful name that describes what will be returned
 - MUST return one value
 - The value is returned through the function name
 - Return value must be used in the calling module
 - Do not use the CALL keyword when calling a function
 - Any required values must be passed in through parameters
 - Must not include INPUT and OUTPUT statements.

Using Modules	Using Functions
<pre> MODULE Main area1 ← 0 INPUT(length) INPUT(height) Call CalculateArea(length, height, area1) END Main MODULE CalculateArea(length, height, area) area ← length * height END CalculateArea </pre>	<pre> MODULE Main INPUT(length) INPUT(height) area1 ← CalculateArea(length, height) END Main FUNCTION CalculateArea(length, height) CalculateArea ← length * height END CalculateArea </pre>
<p><u>Explanation:</u></p> <ul style="list-style-type: none"> • The main module in the above code reads in the length and height so that we can calculate the area. • Note that the module <i>CalculateArea</i> is called using the <i>Call</i> keyword. • <i>length</i> and <i>height</i> are passed into the module as <i>value</i> parameters. That is, their value in the calling module will not change • <i>area</i> is passed in as a <i>reference</i> parameter so that the result can be returned to the main module • NOTE: in pseudocode used in ATAR exams, it is not explicitly stated that a variable is a reference parameter – you need to work this out from the context of the pseudocode. As a rule of thumb, if the value is only used within the module it will be passed by value, if a value needs to be returned to the calling module it will be passed by reference. 	<p><u>Explanation:</u></p> <ul style="list-style-type: none"> • The main module in the above code reads in the length and height so that we can calculate the area. • A <i>function</i> called <i>CalculateArea</i> is then used to calculate the area. • <i>length</i> and <i>height</i> are passed into the module as <i>value</i> parameters. That is, their value in the calling module will not change • The function <i>CalculateArea</i> then calculates the area and assigns the result to the name of the function. • When the function returns a value, that value is assigned to the variable <i>area1</i> in the main module. • NOTE: In pseudocode, functions will usually take in value parameters

Exercises

Bridget was surprised to learn that her school is still recording swim times at the annual swimming carnival with pen and paper. Having done all the programming practice activities for her Computer Science class, Bridget confidently offered to write a program to make the job easier. So far, she's made a good start but needs help finishing it!

Her program uses a record to store information about each swim. She then plans to store each record in an array so that she can easily calculate things such as the average swim time for each house and the year group champions.

Consider Bridget's partially completed algorithm below:

```
RECORD
  SwimData
    studentID: Integer
    yeargroup: Integer
    house: String
    time: Float

GLOBAL
  swims ← SwimData[]

MODULE Main
  CALL EnterSwimTimes()

  INPUT(house)
  OUTPUT(HouseAverage(house))

  INPUT(yeargroup)
  OUTPUT(YearGroupChamp(yeargroup))
END Main
```

Part 1

Write the module `EnterSwimTimes()` such that the user can enter all the relevant data for each swim, including the `studentID`, `yeargroup` and `house` of each swimmer as well as their time.

Your module should:

- Allow for an unknown number of swims be entered.
- Ensure that all times are greater than 0.
- Store all information about each swim in the global array *swims*.

Part 2

Write the function *HouseAverage* that will return the average swim time across all swimmers in a given *house*.

Part 3

Write the function *YearGroupChamp* that will return the *studentID* for the student with the fastest swim time in a given *yeargroup*.

Desk Checking

- modelling of an algorithm using trace tables to test for logic
- apply the following algorithmic and programming techniques:
 - select and apply suitable test data and testing techniques
 - use trace tables to test for and debug logic errors

- After creating an algorithm, it needs to be tested for logic errors to ensure that it will work as expected. One common way of doing this is to use a **trace table**

Completing a Trace Table

1. Select test data that will test **all** possible paths through your program
2. Calculate what the expected output from your code should be for each piece of test data
3. Draw a table with all relevant variables and conditions
4. Walk through the algorithm using each piece of test data
5. Record the values for each variable/condition as you step through the code

Example

Pseudocode	Trace Table			
1 MODULE Main 2 INPUT(x) 3 WHILE x < 18 4 x ← x + 7 5 END WHILE 6 OUTPUT(x) 7 END Main	Line	X	X < 18	OUTPUT
	2	5		
	3		TRUE	
	4	12		
	3		TRUE	
	4	19		
	3		FALSE	
	6			19

Exercises

Question 1

Consider the following pseudocode:

```

1  MODULE Main
2      min ← 100
3      total ← 0
4      average ← 0
5      FOR i ← 1 TO 3 DO
6          INPUT(num)
7          total ← total + num
8          IF min > num THEN
9              min ← num
10         END IF
11     END FOR
12     average ← total / 5
    
```

13 END Main

Complete a trace table for the algorithm using the following test data for the variable *num*:

5 3 7

Line	Min	Total	Average	i	num	Min > num

Question 2

Refer to the following code snippet to answer the questions below.

```

BEGIN
  n ← "Jeff"
  found ← False

  FOR i ← 0 TO 4 DO
    IF names[i] = n THEN
      found ← TRUE
      OUTPUT("Found")
    END IF
  END FOR

  IF found = False THEN
    OUTPUT("Not found")
  END IF
END

```

Use the following test data to complete the trace table for the algorithm in the table below.

names ← ["Chris", "Ashley", "Graham", "Jeff", "Geoff"]

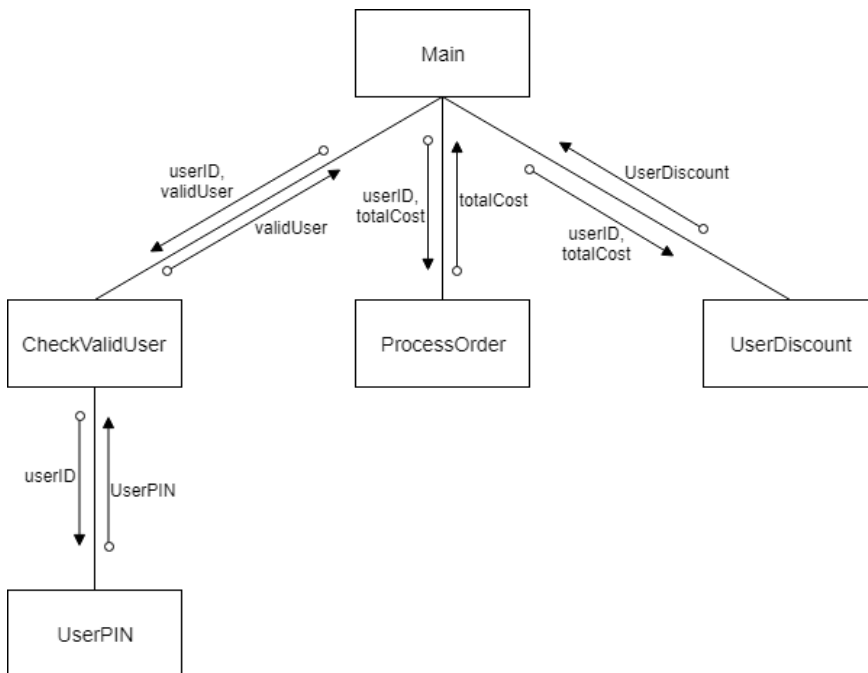
n	found	i	names[i] = n	Output	found = False

Structure Charts

- software development documentation as a part of the SDC
 - structure charts using the Yourdon and Constantine method
- apply the following algorithmic and programming techniques:
 - interpret and create structure charts with parameter passing

- A structure chart is used to graphically show the modules in a program and how they are related.
- Structure charts show how parameters are passed between modules
- The symbols used in a structure chart are:
 - A rectangle is used to represent each module
 - Each module is connected to its calling module with a line
 - Parameters are shown with an arrow with an open circle at one end
- The steps to draw a structure chart are:
 1. Identify the main module
 2. Identify other modules
 3. Identify the parameters that are passed between the modules

Example 1



Example 2

Convert the following pseudocode to a structure chart.

Pseudocode

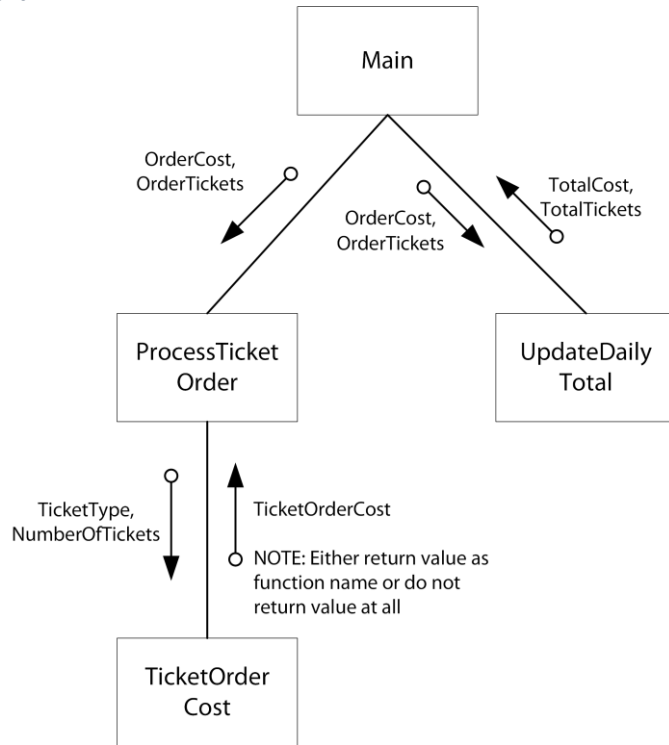
```
MODULE Main
  TotalCost ← 0
  TotalTickets ← 0
  INPUT(MoreCustomers)
  WHILE (MoreCustomers = TRUE)
    CALL ProcessTicketOrder(OrderCost, OrderTickets)
    CALL UpdateDailyTotals(OrderCost, OrderTickets, TotalCost, TotalTickets)
    INPUT(MoreCustomers)
  END WHILE
END MODULE

MODULE ProcessTicketOrder(OrderCost, OrderTickets)
  OrderCost ← 0
  OrderTickets ← 0
  INPUT(MoreTickets)
  WHILE (MoreTickets = TRUE)
    INPUT(TicketType)
    INPUT(NumberOfTickets)
    OrderCost ← OrderCost + TicketOrderCost(TicketType, NumberOfTickets)
    OrderTickets ← OrderTickets + NumberOfTickets
    INPUT(MoreTickets)
  END WHILE
END ProcessTicketOrder

FUNCTION TicketOrderCost(TicketType, NumberOfTickets)
  CASE TicketType OF
    Child: Cost ← 0
    Student: Cost ← 5
    Adult: Cost ← 12
    Pensioner: Cost ← 7
  END CASE
  TicketOrderCost ← NumberOfTickets * Cost
END TicketOrderCost

MODULE UpdateDailyTotals(Cost, Tickets, TotalCost, TotalTickets)
  TotalCost ← TotalCost + Cost
  TotalTickets ← TotalTickets + Tickets
END UpdateDailyTotals
```

Structure chart:



Exercise

Consider the pseudocode below.

```
GLOBAL CONSTANT
    WEEK_HOURS ← 37.5

MODULE Main
    name ← ""
    base_rate ← 0
    hours ← 0

    INPUT(name)
    INPUT(base_rate)
    INPUT(hours)

    gross_pay ← CalculateGrossPay(base_rate, hours)
    tax ← CalculateTax(gross_pay)
    FinalisePayment (gross_pay, tax, name)
END Main

FUNCTION CalculateGrossPay(rate, hours)
    IF hours > WEEK_HOURS THEN
        overtime_pay ← CalculateOvertimePay(rate, hours – WEEK_HOURS)
        CalculateGrossPay ← WEEK_HOURS * hours + overtime_pay
    ELSE
        CalculateGrossPay ← hours * rate
    END IF
END CalculateGrossPay

FUNCTION CalculateTax(pay)
    total_tax ← 0
    CASE pay OF
        pay <= 500:
            total_tax ← 0
        pay <= 1000:
            total_tax ← pay * 0.25
        pay > 1000:
            b2 = (pay – 1000) * 0.35
            b1 = 1000 * 0.25
            total_tax ← b1 + b2
    END CASE
    CalculateTax ← total_tax
END CalculateTax

FUNCTION CalculateOvertimePay(base_rate, overtime_hours)
    CalculateOvertimePay ← 0
END CalculateOvertimePay

MODULE FinalisePayment (pay, tax, name)
    PRINT("Payment finalised")
END FinalisePayment
```

Create a structure chart to reflect the pseudocode.

Networks and Communication

Terminology

The following are some general networking terms that you should know:

- Protocol
- Bandwidth
- Broadcast traffic
- Transmission medium
- Data collision
- Attenuation
- Latency
- Bit rate

Network Devices

- role of the following hardware devices in network communications:
 - router
 - switch
 - firewall
 - modem
 - network interface card (NIC)
 - wireless access point
 - bridge
 - gateway
 - repeaters

Device Roles

- **Router:**
 - A router is a network device that connects two or more networks together and directs packets of data between them. Routers form an integral part of the Internet as they determine the best path for each packet of data to take to reach its destination.
- **Switch:**
 - A switch joins multiple devices together to form a single local area network (LAN). A switch stores the MAC address for each device that is connected to it, so when it receives a piece of data (called a **frame**), it can determine which port it needs to direct the frame to.
- **Firewall:**
 - A firewall is a device that restricts traffic both in and out of a network and helps to prevent unauthorised users gaining access to a network. It can operate either as a hardware device or as a software program (or both).
- **Modem:**
 - A modem (or **modulator/demodulator**) is a device that allows networks to connect to each other over the phone lines by converting data from digital to analogue signals and vice versa.

- **Network interface card (NIC):**
 - A Network Interface Card (NIC) allows a device to connect to a network using a specific transmission medium. For example, an Ethernet card allows a twisted pair cable to plug into the device and communicate using the Ethernet standard. A device needs a NIC for each transmission medium or network standard that it uses. That is, it needs a different NIC for ethernet, wireless and Bluetooth.
- **Wireless access point (WAP):**
 - A wireless access point (WAP) allows devices to connect to a LAN without the need to physical cables. The WAP connects to a router or switch using wires, and then projects a Wi-Fi signal that other devices can use to connect to the network.
- **Bridge:**
 - A bridge is a device that connects two LAN segments together and treats devices on either side of the bridge as if they are all on the same LAN. That is, it creates a single, larger LAN. This is different from a router in that a router keeps each network it connects to separate.
- **Gateway:**
 - A gateway is a network node that serves as an access point to another network. This can involve a change of addressing or different networking technology. Routers and other computers such as servers are often used as gateway devices.
- **Repeaters:**
 - Repeaters help to extend the range of transmissions by retransmitting a signal they receive to devices farther away that might otherwise not be able to receive a transmission due to signal loss.

Transmission Media

- characteristics of wireless transmission media, including:
 - broadcast radio
 - satellite
 - microwave
 - cellular
 - characteristics of wired transmission media, including:
 - twisted pair (unshielded twisted pair [UTP] and shielded twisted pair [STP])
 - fibre optic (single-mode, multi-mode)
- A transmission medium is the way that we physically transfer data from one device to another.
 - There are two main types of transmission media:
 - **Wired:** data travels over a physical connection between two devices
 - **Wireless:** data travels through the air between devices, which are not physically connected to each other

Wireless Transmission Media

- **broadcast radio** - used for Wi-Fi and Bluetooth
- **satellite** - space station receives microwave signals and transmits them back to Earth
- **microwave** - uses microwave signals for high-speed transmission, but must have line-of-sight between transmission towers
- **cellular** - used for mobile communications such as wireless modems and phones

Wired Transmission Media

Twisted Pair

- Thin copper wire arranged in pairs that are twisted around each other to minimise interference
 - **Unshielded Twisted Pair (UTP)**: insulated cables together in single cable
 - **Shielded Twisted Pair (STP)**: pairs have foil shield around each pair, and then an overall shield

Fibre Optic

- Thin strand of glass (or plastic) that transmit digital data using light
- Uses total internal reflection to bounce light down the glass strand

Wired vs Wireless

Wired	Wireless
<ul style="list-style-type: none"> • Increased security as need to physically connect to wire to access signal • Usually, greater distances can be covered • Usually, higher bandwidth 	<ul style="list-style-type: none"> • Cannot prevent people from intercepting signal, therefor can be less secure • More susceptible to atmospheric conditions • Gives greater flexibility • Requires less infrastructure as don't need cables, switches for each device

Exercises

Question 1

Jacob lives on a farm in regional WA and would like to get access to broadband Internet. There is currently an old copper phone line running to the property that he has been using for a dial-up Internet connection. In the distance he can see a local mountain with a telecommunications tower that can transmit both cellular and microwave signals.

With reference to the most appropriate transmission medium(s), explain how Jacob could get access to a high-speed broadband Internet connection throughout his house. Provide justification for your choices.

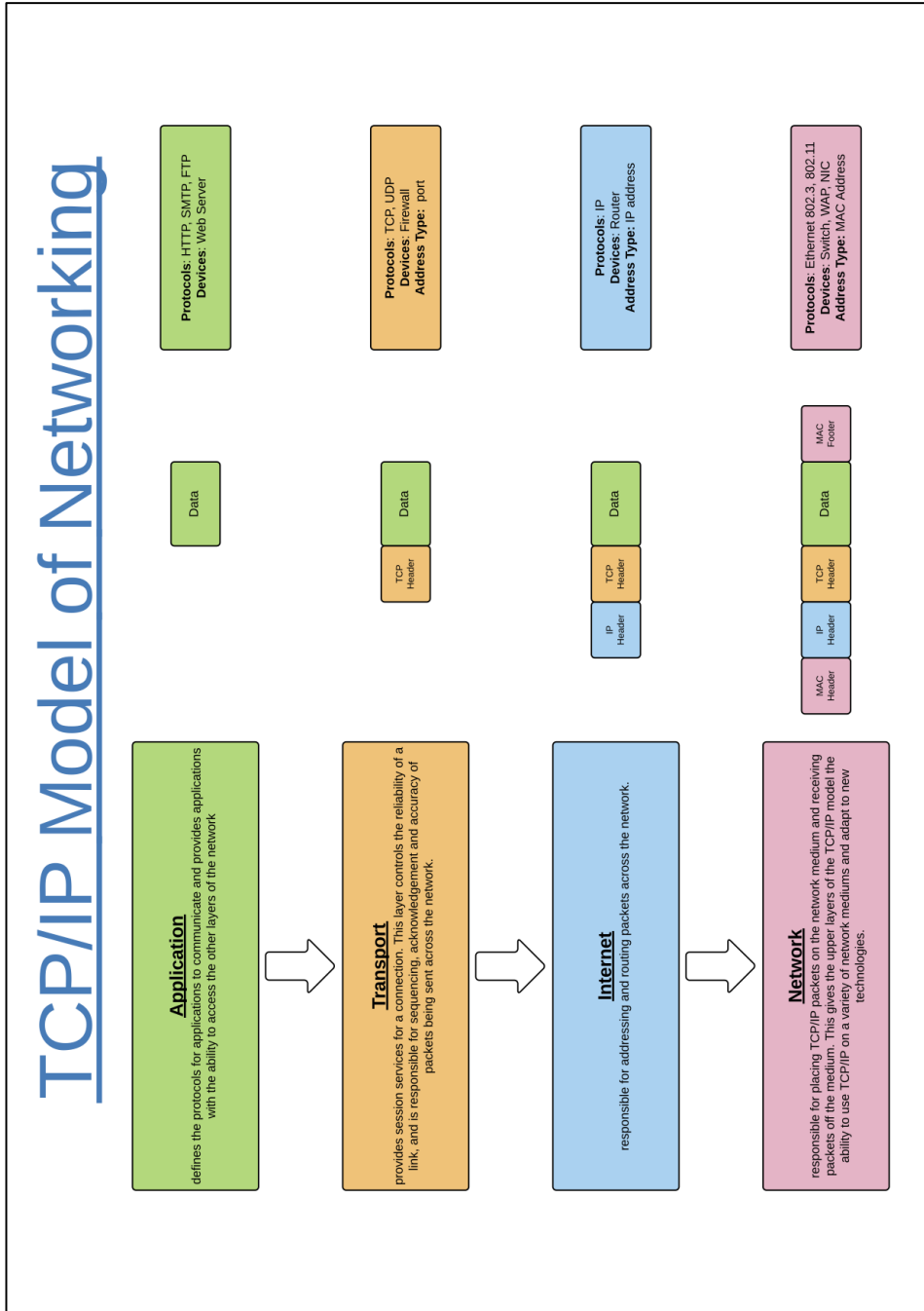
TCP/IP Model

- role of the layers within the Department of Defence (DoD) transmission control protocol/internet protocol (TCP/IP) model in network communications
- purpose of the layers within the DoD TCP/IP model, including:
 - application layer
 - transport layer
 - internet layer
 - network layer

- The TCP/IP model of networking is a framework that describes how a network is set up and how data can be transferred across the network.
- One of the key advantages of using the TCP/IP model is that it ensures that people using different systems are still able to communicate. If they follow the same rules, it does not matter what software or TCP/IP stack is used.
- Each layer can operate independently. That means that as long as each layer can interact with the other layers using the protocols that have been defined, it does not matter how that layer completed its job. For example, as long as the network layer passes IP packets to the Internet layer, it does not matter what technology the Network layer uses to pass the data from one place to another.

TCP/IP Layers

- There are four layers to the TCP/IP model:
 - Application
 - Transport
 - Internet
 - Network
- NOTE: You will see these layers named differently in various textbooks and websites. Make sure you **use the layer names that are specified in the syllabus**.
- Each layer is responsible for a specific aspect of network communications. See the graphic below to see what each layer does
- Different protocols operate at different layers and define how the network should perform different functions.
- Learn what the different protocols do and what layer they operate at. This will help you understand the purpose of each layer.



Exercises

Question 1

For each layer of the TCP/IP model, identify one protocol and use that to describe to purpose of that layer.

Layer	Protocol	Purpose
Application		
Transport		
Internet		
Network		

Question 2

Complete the table below with information about various devices used within a network. Describe the role of the device, which layer of the TCP/IP model it operates at and what type of addressing it uses.

Device	Role	TCP/IP Layer	Address Type
Switch			
Router			
NIC			
Firewall			

Question 3

TCP and IP are two important protocols that form the basis of most Internet communications.

(a) Explain the purpose of TCP.

(b) Explain the purpose of IP.

Network Security

- methods used to ensure the security of networks, including use of:
 - firewalls
 - anti-virus software
 - password and network user policies
 - authentication
 - encryption
- strategies used to compromise the security of networks, including:
 - denial of service
 - back doors
 - IP spoofing
 - phishing

- To keep network communications secure, it is necessary to understand what some of the threats to network security are and methods to keep a network secure.

Security methods

- **Firewalls**
 - A firewall prevents unauthorised access to a network
 - Monitors all incoming and outgoing traffic to determine if it should be allowed or blocked
 - Can be hardware, software, or both
- **Anti-virus software**
 - Prevents viruses being installed on the computer
 - Virus can destroy files and affect performance of a network
 - Once on a network, it can spread through the network
- **Password and network user policies**
 - used to control:
 - Access to a network
 - How often people need to reset passwords
 - Type of passwords that are acceptable
 - User privileges on a network
 - Restrict access to sections of the network
- **Authentication**
 - Authentication is the process of verifying a user is who they say they are
 - Methods of authentication include:
 - Username and password
 - Biometric authentication
 - Two factor authentication
- **Encryption**
 - Prevents unauthorised parties being able to access confidential information
 - Data is encoded into a form that cannot be understood
 - Data is the decoded using the appropriate decryption key
 - Unencrypted data is also known as plain text data

Network Threats

- **Denial of service**
 - Purpose is to prevent access to network resources
 - Common methods include:
 - Send large number of requests to a web server
 - Send many (or large size) emails to an account
 - Servers cannot process all the requests so the system crashes or runs slowly
- **Back doors**
 - Method to access a computer and/or network that bypasses the normal security measures
 - Sometimes created by developers for testing/maintenance of systems
 - Can also be installed through malware to allow attackers to exploit the system
- **IP spoofing**
 - Attackers can hide their identity and gain access to a network by pretending they are from a trusted source
 - IP packets are modified so that the source IP address matches a trusted IP source
- **Phishing**
 - Uses social engineering to trick a person into giving out personal information
 - Usually involves attacker sending an email, SMS or social media message asking to confirm personal details or click on a link

Exercises

Question 1

Vikki is responsible for network security at a local hospital and has decided to implement new password and network user policies to help ensure the security of the network. Describe three policies that could be implemented to help keep the network secure.

i _____

ii _____

iii _____

Question 2

Explain each of the following network vulnerabilities:

Denial of Service (DoS) _____

IP Spoofing _____

Question 3

(a) Explain the term 'backdoor' as it applies to network security.

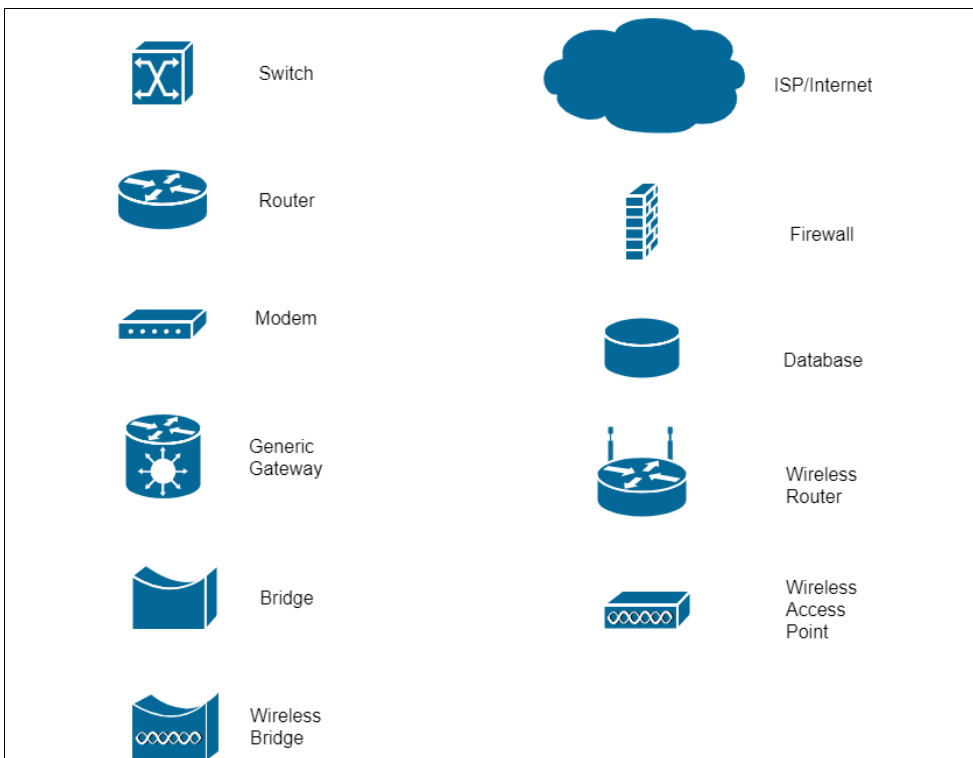
(b) Describe a situation where using a backdoor could be considered ethically sound.

Network Diagrams

- create network diagrams using the CISCO network diagrammatic conventions to represent network topologies for LAN, WLAN and WAN

- A network diagram shows the logical design of a network and how the devices should be connected.
- Generally, the order of devices would be:
 - ISP/Internet/Cloud
 - Modem (if required)
 - Firewall
 - Router
 - Switch
 - WAP / Servers / End devices (such as desktop computers and laptops)
- The WA ATAR course requires the use of CISCO symbols when drawing a network diagram.

CISCO Symbols

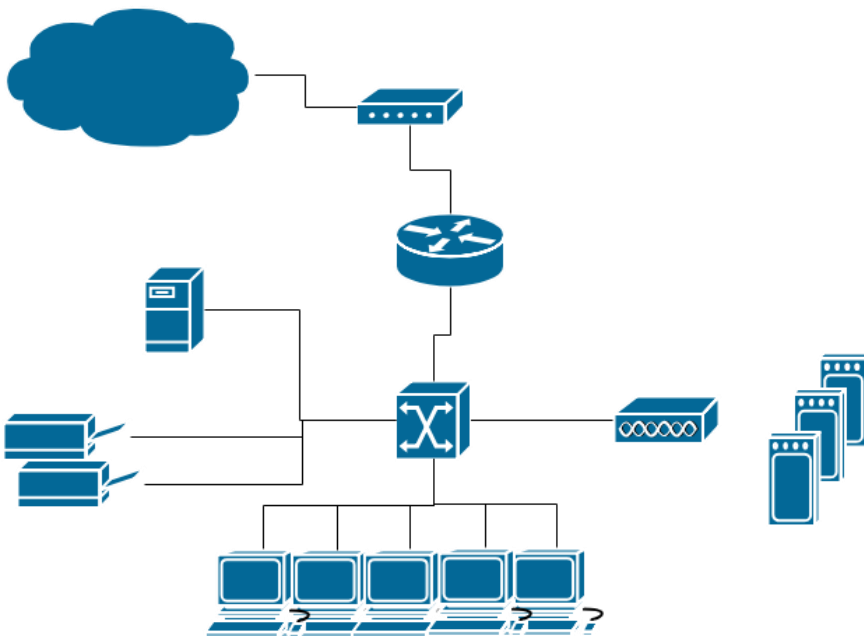


Example

SnakeOil Cars has decided that it is time to modernise their systems and build a new computer network. Their new network will need the following features:

- Internet access via a broadband router with a built-in firewall.
- A combined File and Database server located in the electrical room
- Two network aware photocopiers, one located in the service reception area and one in the hallway.
- UTP connections for 5 desktop workstations used by the receptionist, Service Counter, Service Manager, Finance & Insurance consultant and Vehicle Sales Manager.
- Three tablet computers used by the Sales consultants. The consultants will use the tablets to show customers available options and check vehicle stock as they are dealing with customers.

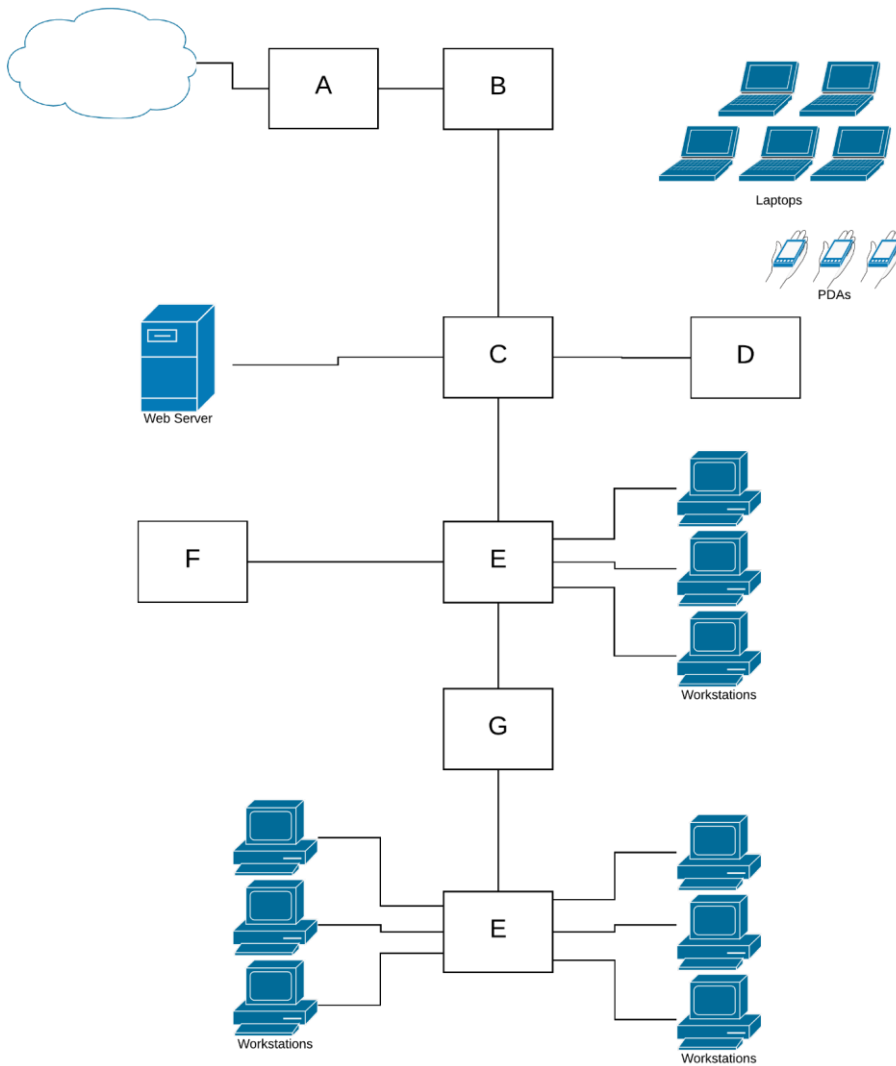
Using CISCO conventions, draw a network diagram to show how SnakeOil Cars should set up their network. You do not need to show the room layout of on your diagram, only the devices and transmission media.



Exercises

Question 1

Whilst building their new system, Furclass have also decided to upgrade their network infrastructure. The following network diagram shows how their new network has been set up.



(a) In the table below, identify the network devices in the diagram that have been labelled A-G. Explain the purpose of each device in a network.

Label	Device	Purpose
A	Modem	
B		
C		
D	WAP	Allows devices to connect to the network using Wi-Fi.
E		
F	NAS	
G		

(b) After setting up the new network, the staff have found that all the workstations are able to connect to the Internet, however nobody can connect via their mobile devices when they are connected to the network. Identify which device (A, B, C, D, E, F or G) is likely to be experiencing problems and explain why you have come to this conclusion.

Device: _____

Reason: _____

Question 2

The warehouse has an office located at the front of the building where a receptionist can greet customers and coordinate the work from a desktop workstation. The database server is located in a locked storage room at the rear of the office, and there is a shared network printer in the corner. There is also a small office where the warehouse manager has a desktop workstation from where she can access the network. The entire network has access to the Internet through a shared connection to their ISP and is protected by a firewall. d) Draw a network diagram to show how the various devices are connected, clearly labelling each device. Label any necessary transmission media, network communications standards and/or network control protocols that would be used with the various parts of the network.

Question 3

Vikki is the network administrator at White Wilderness – an adventure company specialising in family tours to Antarctica. They have fitted a cruise ship out with a computer network to provide guests and crew with Internet access and computing facilities whilst away. The ship's network consists of the following:

- Six computers on the main deck floor forming their own LAN segment where guests can get Internet access and process their photos
- Several handheld devices that visitors can use on the vessel to access email and view video of the local wildlife
- Two computers in the reception desk for staff to process guests when they board the ship and keep track of any special guest requirements
- Three computers in the office area for the crew
- Three laptops that the ship's officers use to connect to the network wirelessly
- One printer that all crew connected to the network can use

Exam Technique

Sitting the Exam

Reading time

- Read scenario in Source Booklet
- Skim over questions in Extended Response section
- Go back to start of paper and skim over short answer questions

Start paper

- Work through short answer questions
- Keep an eye on the time
- Start with questions on topics you are confident with
- Read question carefully:
 - Take note of key words – identify/state/describe/explain
 - Consider marks allocation
 - Answer question that is being asked
- Extended response questions:
 - Read source booklet carefully
 - Follow process to create context and DFD diagrams
 - Use correct symbols for diagrams
 - Use standard syntax and key words for pseudocode
 - Use CISCO symbols for network diagrams. It is also a good idea to label your diagram in case you get the CISCO symbols wrong. That way you will still at least get some marks.
- NOTE: Wherever possible use keywords and terminology from the syllabus to answer questions

Preparation

- Identify your areas of weakness
- Memorise key concepts and terminology. Make sure you know the correct terminology to use.
- Learn the different types of question and the expectations for answering each question
- Practice reading and writing pseudocode
- Complete past papers under timed conditions. If you do not have a solid three-hour block, then complete each section in the suggested time frame

Congratulations! You have now completed your revision booklet!

Edith Cowan University would like to wish all students the best of luck with their future exams!

