2020 HSC Study Day Series

AURORA COLLEGE

HSC Software Design & Development

HSC 2020

NSW Department of Education

www.aurora.nsw.edu.au

2020 HSC Study Day Series

AURORA COLLEGE

Details

Date: Tuesday 23rd June, 2020

Time: 8:50am – 3:10 pm

Location: Adobe Connect room https://connect.schools.nsw.edu.au/aurora-hsc-study3/

Materials: Available to download via this Dropbox link

Recordings: The sessions will be recorded and accessible for registered participants after the event via the same Dropbox link above. These recordings will be accessible until the HSC exam.

Program

Time	Session	
8:50 – 9:00am	**Welcome**	
9:00 – 10:00	**Development and Impact of Software Solutions** *Sarah Duncan, Carlingford High School*	
10:05 – 11:05	**Software Development Cycle: Part 1** (Defining and understanding the problem, Planning and designing software solutions) *Ray Montalban, Arthur Phillip High School & Aurora College*	
11:05 – 11:30	**Morning tea break**	
11:30 – 12:30	**Software Development Cycle: Part 2** (Implementing software solutions; Testing and evaluating software solutions; Maintaining software solutions) *Sarah Duncan, Carlingford High School*	
12:35 – 1:15	**Exam tips and moving up a mark range** *Anil Warrier, Rose Bay Secondary College*	
1:15 – 1:55	**Lunch break**	
1:55 – 3:00	**OPTION: Programming Paradigms** *Anil Warrier, Rose Bay Secondary College* **Adobe room** https://connect.schools.nsw.edu.au/aurora-hsc-study3/	**OPTION: Interrelationship of Hardware & Software** *Ray Montalban, Arthur Phillip High School & Aurora College* **Adobe room** https://connect.schools.nsw.edu.au/aurora-hsc-study2/

Image on front cover attribution: Sourced from http://fsktm.uthm.edu.my/v4/index.php/offered-programmes/postgraduate-programmes/master-of-computer-science-software-engineering

2020 HSC Study Day Series

AURORA COLLEGE

Setting up Adobe Connect

Teachers will need:

- A good, stable Dept of Ed internet connection using an ethernet cable (wifi not recommended)
- Data projector
- Speakers

The sessions will be held via Adobe Connect. Please ensure there is only one connection per school. The presentation can be displayed on a data projector through any computer with an ethernet cable and speakers. The information below will help with setting up if you are not familiar with Adobe Connect.

- You will need to perform all necessary setup in advance of your online session so that you have time to resolve any connection or access issues. The Adobe room will be opened 30 mins prior to commencing to allow time for set up.
- Test your computer prior to accessing your online room by going to the Meeting Connection Diagnostic. Ensure you install any add-ins, if prompted to do so by the connection test.
- The following guide may also be useful Quick Start Guide for Participants.

Entering the Adobe room

Teachers log in once for their class. Students are NOT to log in individually. To enter your online room, click on the Adobe Connect link provided above. Enter by typing in your Department of Education ID (eg: *jane.citizen@detnsw)* in the *Username* field then your DoE password in the *Password* field. The first thing you should do when you enter the room is complete the audio setup wizard. ('Meeting' drop down menu-> Audio Setup Wizard)

For technical help:

If you are having any issues with technology, please contact the Aurora College IT Support Team on 1300 610 733 or support@aurora.nsw.edu.au

Rights and responsibilities

Duty of care for students throughout the day remains with the registered schools and their respective teachers. Please ensure adequate supervision is provided during the day. Respectful and active participation in the event is strongly encouraged through the 'chat' pod.

Evaluation

Constructive feedback is essential, links to online surveys will also be distributed during and shortly after the event. There are two surveys and they both close on 21st September:

- Teachers https://www.surveymonkey.com/r/HSCSTUDYDAYSTEACHER2020
- Students https://www.surveymonkey.com/r/HSCSTUDYDAYSSTUDENT2020

We look forward to your participation.

Software Design and Development HSC Study Day

Development & Impact of Software Solutions

PRESENTER: SARAH DUNCAN
CARLINGFORD HIGH SCHOOL

FINDING PRACTICE QUESTIONS

- Access a HSC paper's marking guidelines
- Scroll to the end and view the mapping grid

Software Design and Development HSC marking guidelines 2019
719 KB, 22 Pages

2019 HSC Software Design and Development Mapping Grid

Section I

Question	Marks	Content	Syllabus outcomes
1	1	9.1.2 Approaches	H1.2
2	1	9.2.4 Volume testing	H4.2
3	1	9.1.2 Outsourcing	H1.2
4	1	9.2.4 Post-implementation review	H5.2
5	1	9.1.2 CASE tools	H1.2
6	1	9.2.2 Data dictionary	H5.2
7	1	9.2.2 Search/complete the algorithm	H4.2
8	1	9.2.3 Use of flags	H4.2
9	1	9.2.3 Compilation/interpretation	H4.2
10	1	9.2.2 Control structure (convert post-test to pre-test)	H4.2
11	1	9.1.2 Methods of installation	H1.2

SOCIAL & ETHICAL ISSUES

- Issues **don't** always mean **bad** …
- It means there are some elements that may need more thought before beginning and some extra features or security
- Try to include in your response one **positive** comment before you discuss all the negatives.
- While the system is intuitive for skilled users, the developers must consider older generations whose skills may limit their use of the software as well as users with visual limitations.
- Discuss, Evaluate, Explain, Analyse they all mean look at different sides.

Steps for preparing/studying

- Write an answer to a HSC question in a time limit
- Find the marking guidelines and look at the marking criteria – what mark would you give your answer?
- Read the sample answer and find any key terminology that you didn't include or find what your response was missing – an example or more than 1 advantage/disadvantage (keep your eye out for plurals)
- Rewrite your response (without looking back at the sample answer but look back at your original response)
- Mark your 2nd response

Develop a list of topics and key terminology or words to use in a question

TOPIC	KEYWORDS/FEATURES
Changing Nature of Work	Job loss Job description changes Retraining
Software Market	4Ps - Price, Promotion, Product, Place Word of mouth
Ergonomics	Work environment, Furniture Interface design – colour/placement/sizes

SOCIAL & ETHICAL ISSUES

- **Preliminary** as well as **HSC** course
- Be able to apply to any given scenario & identify issues in a scenario. Also be able to discuss an issue from the point of view of a developer or a user.
  - Ergonomics
  - Inclusivity
  - Interface Design

  P
  - Impact of software on society & individuals
    - Malware & Viruses, Cyper Safety, Social Networking
  - Rights and responsibilities of a developer
  - Piracy, copyright laws , intellectual property
  - Types of licenses (Commercial, Shareware, public domain, open source), ownership vs licensing, collaboratively developed software
  - Reverse Engineering, Decompilation

Question 21 (9 marks)

A new smartphone app is to be developed for registered users to hire cars. Registered users can use the app to find the location of available cars, unlock them for use and lock them after use. The bank account of the user is debited automatically based on the time for which the car was used. The app will also be continually upgraded to enhance user experience.

(a) Outline the social and ethical issues that the developer needs to take into account when developing this app. **3**

Criteria	Marks
• Outlines social and ethical issues with reference to this system	3
• Outlines one social ethical issue with reference to this system	2
• Identifies a social or ethical issue	1

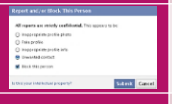
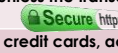
Sample answer:

It is possible to store details of every trip made by each user. This could be used by those wanting to track the routes taken eg employers, stalkers, potential advertisers.

Those not wanting to provide bank account details or without a smartphone or connectivity, can't hire the cars.

Answers could include:

Bank account details need to be stored securely.

CYBER SECURITY ISSUES	What is it?	Why is it an Issue?
LOCATION BASED SERVICES 	Services which allow your current location to be broadcast. E.g Facebook Places/Check-in	Your location can be tracked and monitored. Unknown people may be notified if you are not at home.
UNWANTED CONTACT 	Whilst online you cannot control what information you see and who can contact you.	Information and messages can make you uncomfortable but this cannot be easily controlled
CYBERBULLYING 	Spreading false rumours, teasing, making threats online	Everyone has the right to not be bullied and to feel safe online.
ONLINE FRIENDS 	Making friends online.	Not everyone online is who they say they are. Private information is at risk and one should never meet with an online friend by themselves.
DIGITAL FOOTPRINT 	Every image, video, comment, tweet you place online.	It is virtually impossible to remove information completely from the net. Information is so easily spread.
ONLINE PURCHASING 	Buying products and services online	You don't know what the item really is until it arrives and by this time it has been paid for. Payment information should never be entered unless the transaction is secure. 
IDENTITY THEFT 	Criminals using your personal information and pretending to be you.	They can use your credit cards, access your bank account, take out loans in your name. Be very careful of spyware and phishing attempts and secure your information, passwords and details.

Students learn about:	Students learn to:
The impact of software • inappropriate data structures, for example the year 2000 problem • computer malware such as viruses • reliance on software • social networking • cyber safety • huge amounts of information (which may be unsupported, unverifiable, misleading or incorrect) available through the internet	• recognise the effects of software solutions on society • identify the impact of inappropriately developed software on users • identify the effect of the inappropriate use of software on society and individuals

With any issue you will need to consider **WHY** it is an issue, **HOW** it might affect/it has affected the people or purpose, **WHO** it might affect and **HOW** it might be solved

SOFTWARE LICENSES	Covered by Copyright	Copies allowed? If so how many	Modifications allowed?	Reverse Engineering or Decompilation allowed?	Use in other works
Commercial	✓	1	✗	✗	✗
Open Source	✓	✓	✓	✓	✓
Shareware	✓	✓	✗	✗	✗
Public Domain	✗	✓	✓	✓	✓
Freeware	✓	✓ Not for profit	✓ Not for profit	✓ software that is altered must maintain the freeware classification	✓

2018 HSC Q6

Which type of software is NOT protected by copyright laws?

A. Shareware
B. Open source
C. Public domain
D. Creative commons

2017 HSC Q5

Which of the following is a reason for releasing software as open source?

A. To simplify decompilation of the software
B. To allow others to contribute to improving the software
C. To protect the software developer's intellectual property
D. To gather feedback from users about its features and functionality

SOCIAL & ETHICAL ISSUES

- Current & emerging technology to combat piracy
  - Non-copyable data sheets
    - E.g. Where in the World is Carmen Sandiego 1985
  - Use of serial numbers
  - Site licences
  - Instillation of counter on networks
  - Encryption keys
  - Registration codes – most common combat tool
  - Base-to-base authentication
- Use of Networks – developers and users (**COLLABORATION**)
- The Software Market - **4Ps**
- Legal Action Case Studies (national & international)
  - Know what the issue was and the resolution – good to know dates as well
    - RACV vs Unisys
    - Microsoft vs Netscape
    - NSW T Card system
    - Search Engines (Google vs National censorship laws)
    - Metallica vs Napster

2013 HSC Q4

4 Which of the following is most effective in preventing software piracy?

(A) Using a site licence

(B) Using an encryption key

(C) Making a file 'Read-only'

(D) Providing source code instead of compiled code

2013 HSC Q5

5 The Millennium Bug (also known as the year 2000 problem) resulted from the common practice of only storing the last two digits of a calendar year (eg 63 instead of 1963) in order to save memory.

Which of the following is TRUE of the Millennium Bug?

(A) It was a virus, as it affected software worldwide.

(B) It was a logic error, resulting in inappropriate calculations.

(C) It was caused by an inappropriate data structure, requiring a lot of software to be updated.

(D) It was malware, introduced by programmers who were later employed to fix the problem they had created.

2013 HSC Q11

11 A user pays for and installs a software package with a single user licence. Later, the user pays for and installs an upgrade to the package.

What is the user allowed to do with the original version?

(A) Sell it

(B) Archive it onto a DVD

(C) Give it away but not sell it

(D) Use it on a different computer

2013 HSC Q22A)

Pike Pathology provides patients with medical strips to conduct blood tests at home. These medical strips can be inserted into smart phones using special attachments.

The smart phone has an app that:

- processes data captured from the medical strips
- records the blood test data on the patient's phone
- transmits this data to Pike Pathology.

Software on Pike Pathology's computer system will:

- analyse the data and store the result in the database
- alert the patient if a blood test has not been done
- alert the patient's doctor if the results of a test are abnormal
- allow the patient's doctor to access Pike's database to retrieve results.

The doctor then communicates with the patient, if necessary.

(a) Issues relevant to the use of such a system include access to data, privacy, the need for accuracy, and technical issues. **4**

Discuss TWO of these issues with reference to the scenario.

Criteria	Marks
• Provides a good discussion of two issues, clearly related to the scenario	4
• Provides a discussion of relevant issues with reference to the scenario	3
• Discusses a relevant issue OR • Elaborates on two issues	2
• Show some understanding of a relevant issue	1

DISCUSS

- What the issues are
- How they can affect the user/data
- How the app may address any concerns

Privacy: Due to the sensitive nature of the personal and medical data being collected & transmitted by the app it is important to consider which doctors have access to the data. It is also important to ensure that the data, particularly blood test results and diagnosis, is not accessed by unauthorised personnel and transmitted data must be secured against falling into the wrong hands.

The app and attachments to collect data at home are already addressing privacy by eliminating the technicians and drivers from having access to this data. However, issues of secure transmission arise.

Accuracy: It is imperative that the app that is collecting the data, the transmission of results & data as well as saving data in the database is accurate and timely. If the data is inaccurate or the process causes errors to occur then inaccurate data may lead to inaccurate diagnosis and test results and could potentially affect patient health. It is also important to ensure that any and all medical history data is kept accurate to not allow patients to be misdiagnosed or incorrectly prescribe medication or dosages.

2013 HSC Q23

Describe TWO ways in which a team of developers can make use of computer networks to support the development of software. **3**

- FROM THE SYLLABUS:
  - Use of networks [Collaboration]
    - By developer when developing software
      - Access to resources
      - Ease of communication
      - Productivity

Criteria	Marks
• Describes TWO ways in which a team of developers can make use of networks to support software development	3
• Identifies TWO ways in which a team of developers can make use of networks to support software development OR • Describes ONE way in which a team of developers can make use of networks to support software development	2
• Identifies ONE way in which developers can use networks to support software development	1

Sample answer

Access to resources: developers can use the internet to access code libraries, search for solutions for problems that arise, try to find more efficient ways of solving problems etc. For example, a developer may not know how to use a specific function in a programming language, and searching for information on it will show examples of its use.

Ease of communication: developers can use messaging programs to communicate with other members of the programming team who may be located over a wide area. In addition to chat, they can exchange code fragments, screenshots of screen designs etc and gather feedback or ask for assistance.

2014 HSC Q21

Using examples, describe how legal action can result from software development. **4**

Clients suing developers
Copyright laws – code, GUI (Microsoft vs Apple), intellectual property breaches
Breaches against software licence

Criteria	Marks
• Clearly describes how legal action can result from software development • Uses specific examples	4
• Shows some understanding of legal action resulting from software development • Includes example(s)	3
• Shows a basic understanding of legal action relevant to software	2
• Identifies a feature of legal action relevant to software	1

Sample answer:

Software not performing as agreed by the developer and their client could lead to legal action by the client. If the developer refuses to modify the software to perform as expected or agreed, this could result in financial loss to the client.

Legal action can result from developers using copyrighted code from other sources without paying licensing fees or gaining permission for the use of the code. Where a developer uses source code without gaining proper permission from the copyright owner, then the copyright owner has the option of taking legal action against the developer.

In some instances, the copying of the "look and feel" of a competitor's application can also result in legal action being taken by the copyright owner. A recent example of this type of legal action was the law suit between Google and Apple over aspects of the design of the user interface in Android and Apple iPhone applications.

Application of Software development approaches

- Software Development Approaches
- You will need to: know a description, advantages & disadvantages, appropriateness, compare each, justify for a given scenario
  - Structured (Not traditional)
  - Agile
  - Prototyping
  - RAD
  - End User
  - Combinations

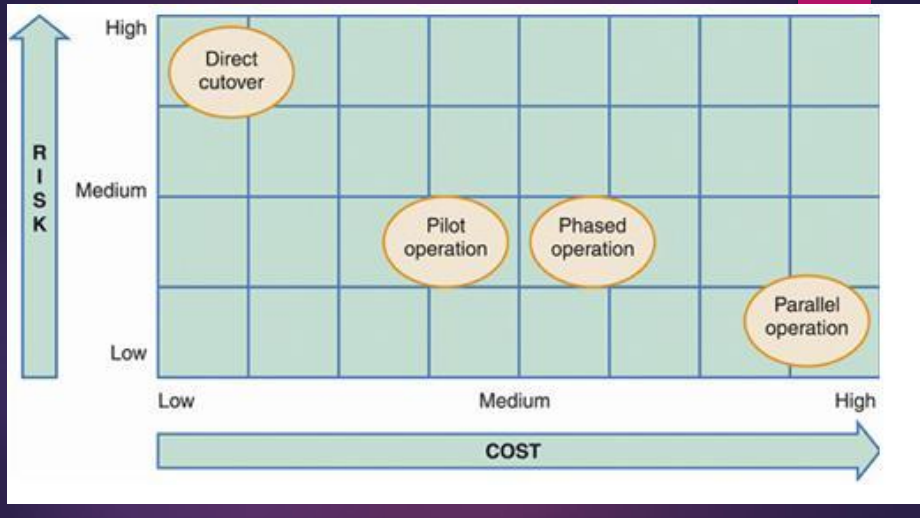
Approach	Advantages	Disadvantages
Structured	VERY STRUCTURED WELL DOCUMENTED	REQUIRES LARGE BUDGET LARGE PROJECT TEAM TIMELY DEVELOPMENT
Agile	MEETS THE REQUIREMENTS OF THE PARTICIPANTS BETTER QUICKER THAN STRUCTURED	MORE REPETITIVE TASKS LESS DISTINCT REQUIREMENTS
Prototyping	ALLOWS USERS TO SEE MODEL OF THE SYSTEM	LITTLE PROCESSING INFINATE AMOUNTS OF VERSIONS
RAD	SHORTER SOFTWARE DEVELOPMENT CYCLE RESULTING IN REDUCED COSTS	SMALLER SCALE PROJECTS
End User	SMALLER COSTS NO NEED FOR DETAILED DOCUMENTATION	LOWER QUALITY PRODUCT MORE COMPRIMISES

Application of Software development approaches

- CASE tools -- Help, Aid, Assist
  - Production of documentation
    - System modelling
    - Data dictionary creation
  - Version Control SW
  - Test Data Generation
  - Production of code

Application of Software development approaches

- Instillation methods of new or updated systems
- You will need to: know a description, advantages & disadvantages, appropriateness, compare each, justify for a given scenario
  - Direct Cut over
  - Parallel
  - Phased
  - Pilot
- Employment Trends in software development
  - Outsourcing
  - Contract Programmers



Application of Software development approaches

- Trends in Development
  - Changing nature of the work environment
  - Changing nature of applications
    - Web-based SW
    - Learning objects
    - Widgets
    - Apps & applets
    - Web 2.0 tools
    - Cloud computing
    - Mobile technology
    - Collaborative environments

2014 HSC Q9

9 Which row of the table best matches an installation method with a valid reason for choosing that method?

	Installation method	*Reason for choosing the method*
(A)	Pilot	The new system is only for a small number of users
(B)	Direct cut over	Training time is needed in the new system
(C)	Parallel	The availability of the data in the system is critical
(D)	Phased	Resources are not available for a proper trial of the new system

2013 HSC Q25A)

A database for a courthouse contains data about the various trials and hearings that will take place. The data includes:

- dates, times and locations
- the names of the presiding judges
- the names of jurors
- details of the accused.

A system is being developed to give stakeholders (eg witnesses, court staff and police) access to appropriate data, using touch screens in kiosks at the entrance to the courthouse as well as web-based access.

(a) Compare the agile approach and an alternative approach, with regard to their suitability for developing this system. **4**

If you see a compare question you can very easily use a table to simply represent your answer

Criteria	Marks
• Provides a comparison of the suitability of the agile approach with an alternative approach, with specific reference to the feature of the approaches that make them suitable for this scenario	4
• Provides a description of the agile approach and an alternative approach, with reference to the suitability for this scenario	3
• Describes a development approach with reference to its suitability for this scenario OR • Describes the agile approach and identifies an alternative approach	2
• Describes a development approach OR • Identifies an alternative approach	1

Structured
~~End User Approach~~
~~Prototype~~
~~RAD~~
Combination

The structured approach will develop a solution entirely and will take longer to develop than the agile approach.

However, because the requirements of the court system are known and there are no known time restraints, the structured approach would be suitable for developing this system. The reliance of the agile approach on the feedback of the stockholders would be somewhat unsuitable as many of the witnesses, police and court staff will be using the system on very few occasions.

A combination of Structured and agile, to create the user interface, could be a suitable compromise for development of the courthouse system.

REASON	AGILE	STRUCTURED
Time	Shorter but with more repetition of the testing and designing stages	Takes longer
Requirements	Less known	More defined and specific
Flexibility	Can add easily when new criteria are though up during testing	

2013 HSC Q26

~~Direct Cut Over~~
Parallel
~~Phased~~
~~Pilot~~

An existing computer-based system is to be replaced. Management wants the new system to be operational quickly. Users need time to be trained in the new system. Many of the users will need to be convinced of the value of the new system. The data generated by the system is very important and must be correct. **3**

Recommend an installation method for this system, and justify your choice.

Criteria	Marks
• Recommends an installation method and provides justification related to the scenario OR • Recognises that the suitability of a method depends on the importance placed on the criteria listed in the question	3
• Describes an installation method	2
• Demonstrates a limited understanding of installation method(s)	1

Sample answer:

Parallel installation would allow the data generated by both the old and new system to be compared, allowing reversion to the old if necessary, and would give users time to adjust and receive training. Direct cutover would not allow training time.

The most appropriate method depends on the importance of the criteria, eg if speed of introduction is considered most important, direct cutover should be used.

OR – Pilot schemes would allow evaluation and could encourage greater user acceptance.

OR – Phased would allow training in parts of the new system.

Software Design and Development

Systems Development Cycle Part 1

Defining and Understanding the Problem
Planning and Designing Software Solutions

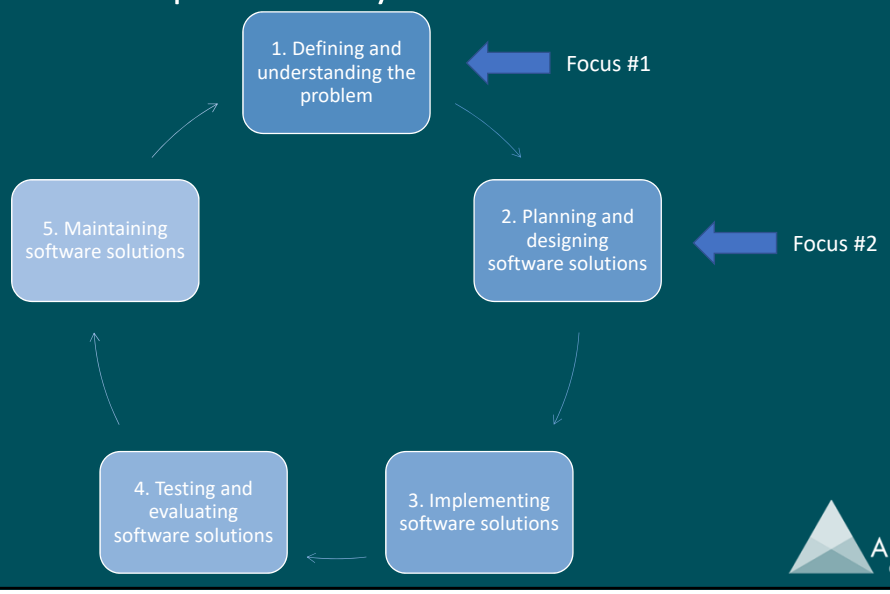
AURORA COLLEGE

Overview

- Systems Development Cycle
  - Purpose
  - Other names and nomenclature
- Defining and understanding the problem
  - Important steps
  - Important diagrams
  - Important documentation
  - Important means of communication
  - Quality assurance
- Planning and designing software solution
  - Algorithms
  - Representation of logic
  - Standard modules or library routines e.g. APIs
  - Software design tools and diagrams
  - Other factors

AURORA COLLEGE

Systems Development Cycle



AURORA COLLEGE

Defining and understanding the problem

AURORA COLLEGE

Defining and understanding the problem

- Defining the problem
- Issues relevant to a proposed solution
- Design specifications
  - Data types, structures and algorithms
  - User interfaces
- Important diagrams: system documentation
  - IPO diagrams
  - Context and Data Flow Diagrams (DFDs)
  - Storyboards
  - Structure charts
  - Systems flowcharts
  - Data dictionaries
- Important communications
- Quality assurance

Defining the problem

Understanding the problem

AURORA COLLEGE

Defining the problem

- We turn what can often be vaguely worded issues from clients into concrete problems which can then be investigated further.

Example

A new international airport will be installing a computer-controlled baggage sorting system. Each piece of baggage will have a bar-coded label attached to it that indicates the destination and class of the passenger. The baggage will move along a conveyor belt until it reaches the chute assigned to its category, into which it will be automatically tipped. At the bottom of the chute, unloaders will pick up the baggage and load it onto trolleys to be transported to the aircraft.

(Software Design and Development HSC Paper, 2001)

AURORA COLLEGE

Defining the problem

Example

A new international airport will be installing a computer-controlled baggage sorting system. Each piece of baggage will have a bar-coded label attached to it that indicates the destination and class of the passenger. The baggage will move along a conveyor belt until it reaches the chute assigned to its category, into which it will be automatically tipped. At the bottom of the chute, unloaders will pick up the baggage and load it onto trolleys to be transported to the aircraft.

(Software Design and Development HSC Paper, 2001)

- What is the problem? Who do you think is the client? What do they want?
  - Functionality: capabilities; what it must be able to do
  - Compatibility: interoperability; what other systems it must work with
  - Performance: rate of processing; how fast or how many operations it must do
  - Boundaries: limits; what parts are and are not looked after

AURORA COLLEGE

Issues relevant to proposed solution

Issue	Examples
Social and ethical issues	• Intellectual property – copyright, licences (e.g. commercial, shareware, freeware) • Ergonomics – effectiveness of screen design, ease of use • Inclusivity – cultural, economic, gender, disability • Privacy
Consideration and customisation of existing software	Type of software application – GUI, CLI, search engine, social networking etc.
Cost effectiveness and licensing	• Commercial licences • Shareware • Freeware – open-source versus regular software
Development approach	• Structured approach • Rapid Application Development (RAD) • Prototyping • Agile • End-User

AURORA COLLEGE

Issues relevant to proposed solution

Example

A new international airport will be installing a computer-controlled baggage sorting system. Each piece of baggage will have a bar-coded label attached to it that indicates the destination and class of the passenger. The baggage will move along a conveyor belt until it reaches the chute assigned to its category, into which it will be automatically tipped. At the bottom of the chute, unloaders will pick up the baggage and load it onto trolleys to be transported to the aircraft.

(Software Design and Development HSC Paper, 2001)

- What social and ethical issues apply to this example?
- What about existing software?
- What type of licence would be applicable to this example?
- What development approach would work here?

AURORA COLLEGE

Design specifications

Viewpoint	Example
Developers	• Data types: integer, Boolean, string, float/double/single, object • Data structures: arrays, records, standard variables, constants • Algorithms: pseudocode or flowcharts
Users	• Interface design: navigation, colours, fonts, consistency etc. • Social and ethical issues: privacy, inclusivity, ergonomics • Relevance

AURORA COLLEGE

Design specifications

Example

A new international airport will be installing a computer-controlled baggage sorting system. Each piece of baggage will have a bar-coded label attached to it that indicates the destination and class of the passenger. The baggage will move along a conveyor belt until it reaches the chute assigned to its category, into which it will be automatically tipped. At the bottom of the chute, unloaders will pick up the baggage and load it onto trolleys to be transported to the aircraft.

(Software Design and Development HSC Paper, 2001)

- What data types, structures and/or algorithms need to be developed to demonstrate understanding of the problem?
- What does the user need to see?

The answers to these questions determine how well the developers have interpreted what is needed.

AURORA COLLEGE

Systems documentation

Input – Process – Output (IPO) Chart

Input	Process	Output
Data that needs to be entered or put into the program	Changes or series of coherent and logically connected steps for the solution to work	End result of processes

AURORA COLLEGE

Systems documentation

Example

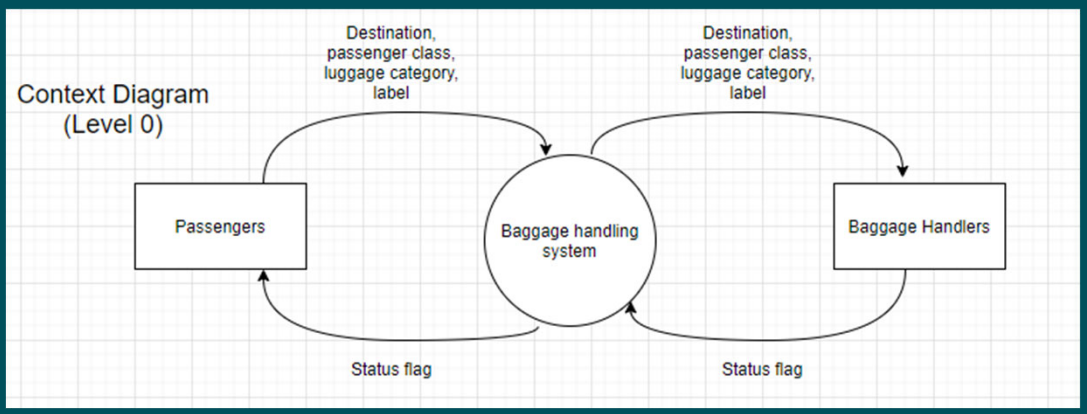
A new international airport will be installing a computer-controlled baggage sorting system. Each piece of baggage will have a bar-coded label attached to it that indicates the destination and class of the passenger. The baggage will move along a conveyor belt until it reaches the chute assigned to its category, into which it will be automatically tipped. At the bottom of the chute, unloaders will pick up the baggage and load it onto trolleys to be transported to the aircraft.

(Software Design and Development HSC Paper, 2001)

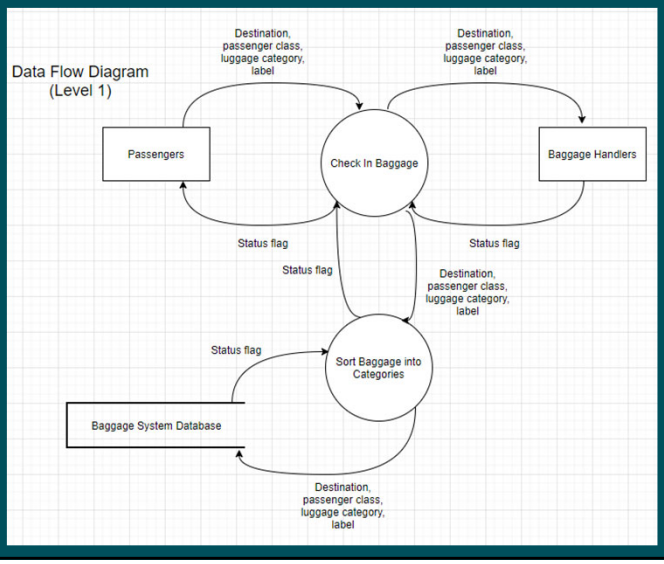
Input – Process – Output (IPO) Chart

Input	Process	Output
Bar-coded label	Bar-coded label on baggage tag matched to flight	Status flag indicating the bag has reached the correct chute
Luggage category	Flight matched to conveyer belt	
Destination	Destination and passenger class indicated on baggage tag matched to trolley	
Passenger class		

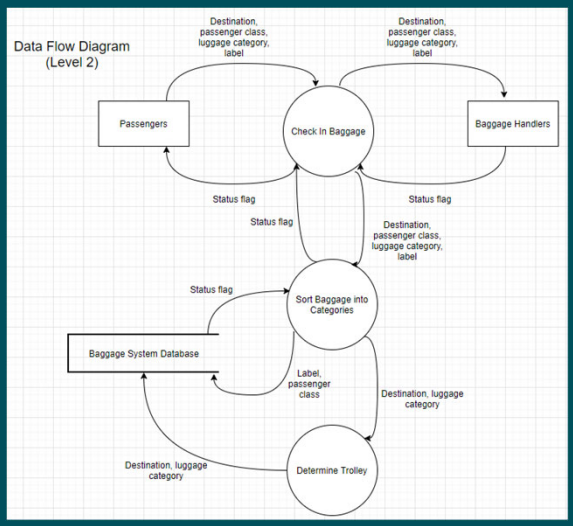
Systems documentation

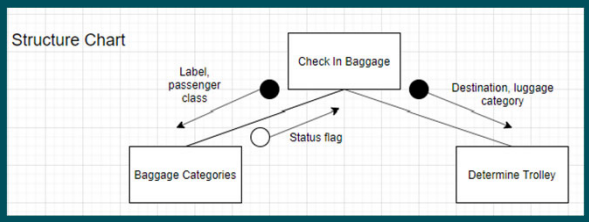


Systems Documentation



Systems Documentation





Note

The IPO Chart has data items that appear on the Context Diagram. These then appear on the two levels of Data Flow Diagram (DFD). These same data items appear on the Structure Chart.

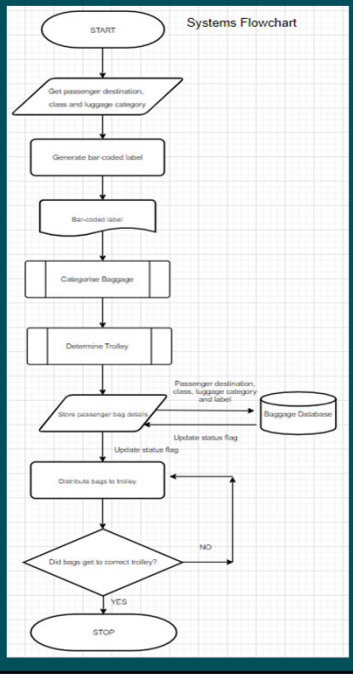
Remember

It is important to see systems documentation as a whole rather than as a part. Each part should 'talk' to each other.

Systems Documentation

Note
Systems Flowcharts should match processes from the DFD where possible, hence “Categorise Baggage” and “Determine Trolley” are taken straight from the DFD on the previous page. Even the data items on the DFD exist on this flowchart.

Remember
It is important to see systems documentation as a whole rather than as a part. Each part should ‘talk’ to each other.



AURORA COLLEGE

Systems Documentation

Data Dictionary

Variable	Data Type	Description
Bar-coded label		
Luggage category		
Passenger class		
Destination		
Status flag		

The ones mentioned in the Design Specifications should be used here e.g. integer, Boolean, string, float/double/single, object etc.

Also note that each variable should correspond where possible to the IPO chart, the Context Diagram, DFD and Systems Flowchart.

AURORA COLLEGE

Important Communications & Quality Assurance

- Open channels of communication between yourself and the client determines project success or failure.
  - Evidence of incorporation of client's perspective
  - Incorporation of client feedback
  - Active involvement of the client
- Open channels allow for explicit definition of quality shared between yourself and the client.
  - Quality criteria: analogous to *success criteria* in other TAS subjects; characteristics, qualities or functionalities which the software must have or meet for it to be deemed successful and of quality.

AURORA COLLEGE

Planning and designing software solutions

AURORA COLLEGE

Planning and designing software solutions

- Standard algorithms
  - Sorts: Bubble, Insertion and Selection
  - Searches: Linear, Binary
  - Maximums and minimums
  - String processing
  - Random number generation
  - File processing
- Custom-designed logic
- Standard modules (library routines)
  - Reasons
  - Examples
  - Issues
- Documentation (systems documentation)
- Interface design
- Factors affecting programming language choice and technology used

AURORA COLLEGE

Standard algorithms

	Insertion sort	Selection sort	Bubble sort
Init:	3 5 1 -1 | 9	3 5 1 -1 9 |	3 5 1 -1 9
1st pass:	3 5 1 | -1 9 (current)	3 5 1 -1 | 9	3 1 -1 5 9
2nd pass:	3 5 | -1 1 9 (current)	3 1 -1 | 5 9	1 -1 3 5 9
3rd pass:		1 -1 | 3 5 9	-1 1 3 5 9

AURORA COLLEGE

Standard algorithms

Insertion sort

```
BEGIN InsertionSort
  Let items = [3, 5, 1, -1, 9]
  Let First = 1
  Let Last = number of elements in the
    array
  Let PositionOfNext = Last - 1
  WHILE PositionOfNext >= First
    Let Next = items (PositionOfNext)
    Let Current = PositionOfNext
    WHILE (Current < Last) AND (Next >
          items(Current + 1))
      'look for the position into
      'which to insert the current name,
      'and shuffle the sorted elements
      'along until we find it.
      Increment Current
      Let items(Current - 1) =
        items(Current)
    ENDWHILE
  Let items (Current) = Next
  'put the current name to be sorted
  'into its correct place
  Decrement PositionOfNext
  'effectively shorten the length
  'of the unsorted portion of the array
  ENDWHILE
END InsertionSort
```

Selection sort

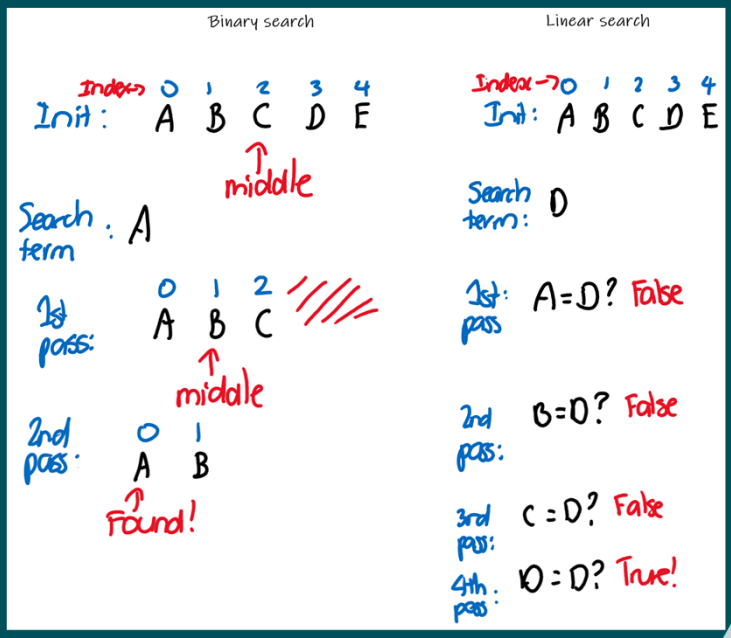
```
BEGIN SelectionSort
  Let items = [3, 5, 1, -1, 9]
  Let EndUnsorted = number of elements
    in items
  WHILE EndUnsorted > 1
    Let i = 1
    Let Max = items(i)
    Let PosMax = i
    WHILE i <= EndUnsorted
      Increment i
      IF items(i) > Max THEN
        Let Max = Name (i)
        Let PosMax = i
      ENDIF
    ENDWHILE
    Swap (items(PosMax), items
          (EndUnsorted))
    Decrement EndUnsorted
  ENDWHILE
END SelectionSort
```

Bubble sort

```
BEGIN BubbleSort
  Let items = [3, 5, 1, -1, 9]
  Let Last = number of elements in items
  Let Swapped = true
  WHILE Swapped = true
      Let Swapped = false
      Let i = 1
      WHILE i < Last
         IF items(i) > items(i+1) THEN
            Swap (items(i), items(i+1))
            Let Swapped = true
         ENDIF
         Increment i
      ENDWHILE
  Decrement Last
  ENDWHILE
END BubbleSort

BEGIN Swap (A, B)
  Let Temp = A
  Let A = B
  Let B = Temp
END Swap
```

Standard algorithms



Standard algorithms

Binary search

```
BEGIN BinarySearch
  Let Name = ['A', 'B', 'C', 'D', 'E']
  Let Lower = 1
  Let Upper = number of elements in the array
  Let FoundIt = false
  Get RequiredName
  REPEAT
    Let Middle = (Upper + Lower) / 2
    Let Middle = integer part of Middle
    IF RequiredName = Name (Middle) THEN
      Let FoundIt = true
      Let PositionFound = Middle
    ELSE
      IF RequiredName < Name (Middle) THEN
        Let Upper = Middle - 1
      ELSE
        Let Lower = Middle + 1
      ENDIF
    ENDIF
  UNTIL FoundIt OR Lower > Upper

  IF FoundIt THEN
    Display "Required name found at " PositionFound
  ELSE
    Display "Required name " RequiredName " not found."
  ENDIF
END BinarySearch
```

Linear search

```
BEGIN LinearSearch
  Let Name = ['A', 'B', 'C', 'D', 'E']
  Let i = 1
  Let FoundIt = false
  Get RequiredName
  WHILE FoundIt is false AND i <= number of names
    IF Name(i) <> RequiredName THEN
      i = i + 1
    ELSE
      Let FoundIt = true
    ENDIF
  ENDWHILE

  IF FoundIt THEN
    Display "Name found at position " i
  ELSE
    Display "Required person not found"
  ENDIF
END LinearSearch
```

AURORA COLLEGE

Standard algorithms

Maximum

```
BEGIN FindMAX
  Let Max = Element (1)
  Let MaxIndex = 1
  Let i = 2
  REPEAT
   IF Element (i) > Max THEN
     Let Max = Element (i)
   Let MaxIndex = i
   END IF
   Let i = i + 1
  UNTIL i > NumElementsInArray
  Display "The highest value is " Max " at position " MaxIndex
END FindMAX
```

Minimum

```
BEGIN FindMIN
  Let Min = Element (1)
  Let MinIndex = 1
  Let i = 2
  REPEAT
   IF Element (i) < Min THEN
     Let Min = Element (i)
     Let MinIndex = i
   END IF
   Let i = i + 1
  UNTIL i > NumElementsinArray
  Display "The smallest value is " Min " at position " MinIndex
END FindMIN
```

AURORA COLLEGE

Standard algorithms

Extract data from strings

```
BEGIN FindDaysandMonths
        Get DateString
        Let StartDays = 1
        Let StartMonths = 4
        'the fourth character in the date string is the start of the month value
        Extract from the StartDays^th character (for 2 characters) from DateString into Days
        Extract from the StartMonths^th character (for 2 characters) from DateString into Month
        Display"the month is " Month " and the day of the month is " Days
END FindDaysandMonths
```

Delete data from strings

```
BEGIN DeleteWordFromString
        Get InitialString, StringToGo
        Let LString = Length of InitialString
        Let Lword = Length of StringToGo
        Let found = 0
        Let i = 0
        REPEAT
                extract from the i^th character (for Lword letters) from InitialString into
                CheckforWord
                IF CheckforWord = StringToGo THEN
                        extract from the 1st character (for i – 1 characters) from InitialString
                        into FirstPart
                        extract from the (i + Lword)^th character (for (LString – Lword – i + 1)
                        characters) from InitialString into SecondPart
                        Let NewString = FirstPart + SecondPart
                        found = 1
                END IF
                i = i + 1
        UNTIL i > = LString OR found = 1
        IF found = 0 THEN
                Display "The word could not be found in your string"
        ELSE
                Display "The new string is " NewString
        ENDIF
END DeleteWordFromString
```

Put new data into strings

```
BEGIN InsertNewWordintoString
        Get InitialString, NewWord
        Let L = Length of InitialString
        Let Found = 0
        Let i = 1
        REPEAT
                extract from the i^th character from InitialString into CheckLetter
                IF CheckLetter = " ; " THEN
                        extract from the 1st character (for i – 1 characters) from InitialString into
                        FirstPart

                        extract from the (i + 1)^th character (for L – i characters) from InitialString
                        into SecondPart

                        Let NewString = FirstPart + NewWord + SecondPart
                        'note that if we use the + operator with strings, the values are concatenated

                        found = 1

                END IF
                i = i + 1
        UNTIL i > = L OR found = 1
        IF found = 0 THEN
                Display "the delimiter ; could not be found in your string"
        ELSE
                Display "The new string is " NewString
        ENDIF
END InsertNewWordintoString
```

Standard algorithms

Random numbers

```
Let R = Random (HighValue – LowValue) + LowValue

BEGIN PrintSixUniqueLottoNumbers
        FOR i = 1 to 99                 'set all flag values initially to zero
                Let Flag(i) = 0
        NEXT i

        FOR i = 1 to 6                  'do this 6 times to generate 6 numbers

                REPEAT
                        Let r = Random (98) + 1
                UNTIL Flag (r) = 0
                'keep generating a random number until one is found which has not been used before

                Let Flag (r) = 1
                'set the flag for this number to show that it has now been used
                Display "Your next number is " r

        NEXT i

END PrintSixUniqueLottoNumbers
```

Standard algorithms

Creating a sequential file

```
BEGIN CreateAFile
    Open FriendsData for output

    FOR i = 1 to 10
        Display "Please enter the details for the next person: "
        Get fname, sname, emailaddr, mobile
        Write FriendsData from fname, sname, emailaddr, mobile
    NEXT i
    Let fname = "xxx"
    Let sname = "xxx"
    Write FriendsData from fname, sname, emailaddr, mobile
    Close FriendsData
END CreateAFile
```

Display data from a sequential file

```
BEGIN DisplayFileContents
    Open FriendsData for input
    Read fname, sname, emailaddr, mobile from FriendsData
    'This is a priming read, performed just before entering the loop to provide the first record (if
    there is one) for printing

    WHILE fname <> "xxx" AND sname <> "xxx"
        Display fname, sname, emailaddr, mobile
        Read fname, sname, emailaddr, mobile from FriendsData
        'this reads subsequent records which can then be tested for the sentinel value before
        they are processed
    END WHILE

    Close FriendsData
END DisplayFileContents
```

Adding data to a sequential file

```
BEGIN AddNewRecords
    Open FriendsData for append
    Display "Please enter the details for the first new person to be added: "
    Display "Enter xxx for both first name and surname to indicate there are no more records to
    be added."

    Get fname, sname, emailaddr, mobile
    WHILE fname <> "xxx" AND sname <> "xxx"
        Write FriendsData from fname, sname, emailaddr, mobile
        Display "Please enter the details for the next new person to be added:"
        Display "Enter xxx for both first name and surname to indicate there are no more
        records to be added"

        Get fname, sname, emailaddr, mobile

    END WHILE
    Close FriendsData
END AddNewRecords
```

AURORA COLLEGE

Standard algorithms

Creating a relative (binary) file

```
BEGIN CreateARelativeFile
    Open ProductData for relative access
    FOR i = 1 to 10
        Display "Please enter the details for the next product: "
        Get ProdNumber, description, quantity, price
        Write ProductData from ProdNumber, description, quantity, price using
        ProdNumber

        'note the use of the variable ProdNumber as the key field, specifying where this
        record will be written in the file.
    NEXT i

    Close ProductData
END CreateARelativeFile
```

Reading data from a relative file

```
BEGIN ReadRecordsFromARelativeFile
    Open ProductData for relative access
    REPEAT
        Display "Please enter the product number for the next product you wish to see:"

        Display "Please enter 999 when you are done"
        Get RequiredProdNumber
        Read ProductData into ProdNumber, description, quantity, price using
        RequiredProdNumber

        'note the use of the variable RequiredProdNumber as the key field, specifying where
        this record will be found in the file

        IF RecordNotFound THEN
            'note the use of the flag RecordNotFound returned by the operating system
            Display "Sorry – no such product"
        ELSE
            Display ProdNumber, description, quantity, price

        END IF
    UNTIL RequiredProdNumber = 999
    Close ProductData
END ReadRecordsFromARelativeFile
```

Updating a relative file

```
BEGIN UpdateRecordsInARelativeFile
    Open ProductData for relative access
    Display "Please enter the product number for the next product whose price you wish to
    update:"
    Display "Please enter 999 when you are done"
    Get RequiredProdNumber
    'priming read in case they wish to exit immediately by entering 999
    WHILE RequiredProdNumber < > 999
        NotFound = 0
        Read ProductData into ProdNumber, description, quantity, price using
        RequiredProdNumber
        'note the use of the variable RequiredProdNumber as the key field, specifying where
        this record will be found in the file

        IF RecordNotFound THEN
            'note the use of the flag RecordNotFound returned by the operating system

            Display "Sorry – no such product"
            NotFound = 1
        ELSE
            Display ProdNumber, description, quantity, price
            Display " Is this the correct product?"
            Get Reply
        END IF
        'do not update until the correct product record is retrieved
        IF NotFound = 0 AND Reply = "Y" THEN
            Get NewPrice
            Write ProductData from ProdNumber, description, quantity, NewPrice using
            ProdNumber

            'update record using data for the new price and the existing data in the other
            fields
        END IF
        Display "Please enter the product number for the next product whose price you wish
        to update:"

        Display "Please enter 999 when you are done"
        Get RequiredProdNumber
    ENDWHILE
    Close ProductData
END UpdateRecordsInARelativeFile
```

Custom-designed logic

- Logic (n): the use of evidence to act as a guide for decisions.
- This refers to the creation of custom algorithms and thus code to do tasks for which algorithms or code does not currently exist or is not up to specification.
  - Consider inputs, processes and outputs
  - Consider algorithmic description method (pseudocode or flowchart), data structures etc.
  - Consider whether you could modify current software to create a customised off-the-shelf package.

AURORA COLLEGE

Standard modules (library routines)

- Consider building up groups of procedures or classes to do common repeated tasks.
- Also known as Application Programming Interfaces or APIs.
  - Why? Efficient, fast, often easier for you.
  - Examples? The Swap() function from the standard algorithms earlier; creating the same form element over and over with different IDs referenceable by code.

AURORA COLLEGE

Documentation

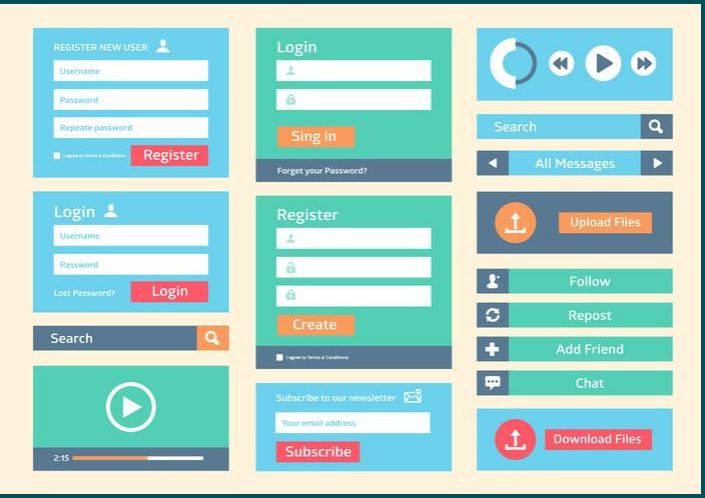
- IPO Charts
- Context diagrams
- Data Flow Diagrams (DFDs)
- Structure Charts
- Storyboards
- Systems Flowcharts
- Data Dictionaries

Notice how these are the same as the ones from Designing and Understanding the Problem?

AURORA COLLEGE

Documentation

Example Storyboard 1



(https://www.vecteezy.com/vector-art/110182-flat-web-user-interface-vector-background)

AURORA COLLEGE

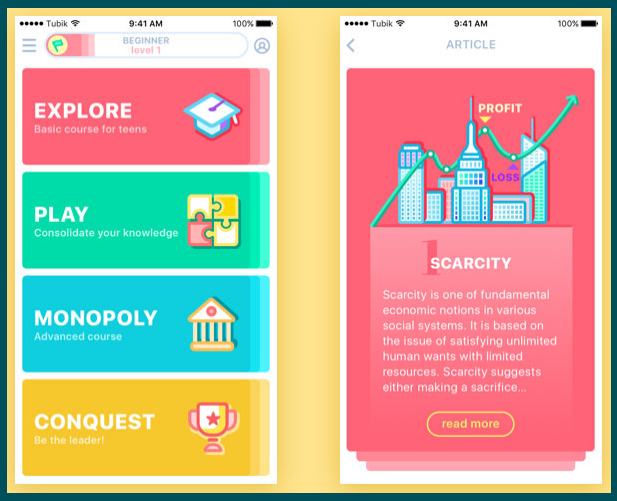
Documentation

Example Storyboard 2

(https://depositphotos.com/79061670/stock-illustration-modern-ui-design-website-elements.html)

AURORA COLLEGE

Interface design



Think about:

1. Audience
2. Screen size
3. Data field size requirements and placement
4. Online help
5. Consistency in approach
6. Recognition of social and ethical issues e.g. inclusivity, privacy, etc.
7. Current and future practices

when designing your software.

AURORA COLLEGE

Programming language choice and technology use

- Sequential languages: statements are executed in the order given.
- Event-driven languages: the computer waits for something to happen and then statements are executed.
- Think about hardware and software use and requirements when choosing programming languages.
- Think about to what extent the hardware and software will need to be pushed when deciding which technologies to consider when writing or delivering a software solution.

AURORA COLLEGE

Software Design and Development HSC Study Day

Software Development Cycle: Part 2
Implementing 9.2.3/ Testing Evaluating 9.2.4/ Maintaining 9.2.5

PRESENTER: SARAH DUNCAN

What you will need to be able to do within each stage:

- Name each step in the cycle and each stage
- Describe specific tasks performed
- Identify and justify the roles of people involved
- Describe, outline or create any documentation produced
- Use the stage specific terminology
- Discuss the use of CASE tools in each step
- Describe the Social & Ethical issues from a scenario

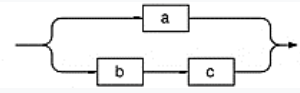
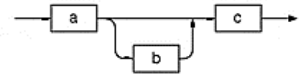
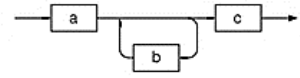
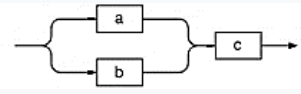
Implementation

- An appropriate Language
- Syntax - Metalanguages
  - EBNF
  - Railroad diagrams
- Translation Process
  - Lexical Analysis
  - Syntactical Analysis
  - Code Generation
  - Compilation Vs Intepretation
- CPU
  - Fetch-Execute Cycle (Fetch/Decode/Execute/Store)
  - Instruction Format
  - Registers, Accumulator, Program Counter
  - Linking & DLLs

Implementation

- Discus, Edit, Add, Fix Code
  - What is good code?
  - Detect & remove syntax and logic errors
  - Runtime errors
  - Software Debugging tools
- Incorporate help
- Implement interfaces
  - WIMP / GUI
  - Navigational aids
- Documentation
  - Developers
  - Using CASE Tools
  - Users
- Subsequent maintenance
- Impact of Emerging Technology

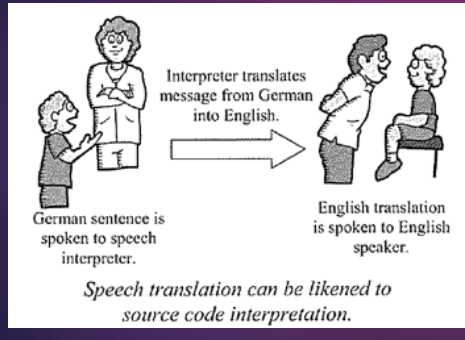
EBNF & Railroad

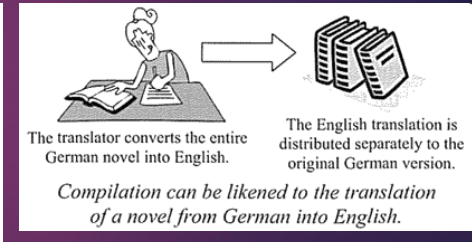
Usage	EBNF Notation	Railroad
Alternation/ or	|	a | b c 
optional	[ ]	a [b] c 
repetition	{ }	a {b} c 
grouping	( )	(a | b) c 

Interpretation Vs Compilation

Each line or statement of source code is translated into machine code and then immediately executed.

All source code is translated into executable code which can be executed at any time





1. Lexical Analysis
2. Syntactical Analysis
3. Code Generation

KEY WORDS FOR COMPILATION

Lexical

LEXICON – LANGUAGE DICTIONARY

RESERVED WORDS

TOKENS

SEQUENTIALLY

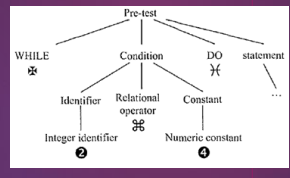
LEXEME – A STRING OF CHARACTERS

TOKENISED

Syntactical

PARSING – GOING THROUGH THE SYNTAX OF A SENTENCE

PARSE TREE



GRAMMATICAL RULES OF A LANGUAGE

COMPATIBLE

ERROR MESSAGES

Code generation

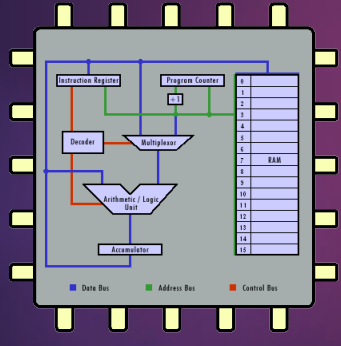
WITHOUT ERROR

CODE IS GENERATED

CONVERTING TOKENS INTO MACHINE CODE INSTRUCTIONS

NO ERRORS FOUND

Fetch Execute Cycle & CPU



CPU INTERACTIVE – Has code samples in machine code & Assembly as well as animated code walk throughs
http://courses.cs.vt.edu/~csonline/MachineArchitecture/Lessons/CPU/index.html

THE CPU's five basic components:

- ***RAM***: Temporary memory which store instructions and values to be used by the CPU
- ***REGISTERS:*** these components are special memory locations that can be accessed very fast.
  - E.g. Instruction Register (IR), the Program Counter (PC), and the Accumulator.
- ***BUSES:*** these components are the information highway for the CPU. Buses carry data between components.
  - E.g. Address, the data, and the control buses.
- ***ALU:*** The number cruncher of the CPU. **Arithmetic/Logic Unit**. Performs all the mathematical calculations of the CPU. Can add, subtract, multiply, divide, and perform other calculations on binary numbers.
- ***CONTROL UNIT:*** this component is responsible for directing the flow of instructions and data within the CPU. The Control Unit is actually built of many other selection circuits such as decoders and multiplexors.
  - E.g. Decoder and the Multiplexor can compose a Control Unit.

2013 HSC Q10

10 Which type of coding error is test data used to detect?

(A) Compiler

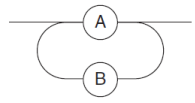
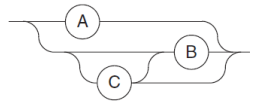
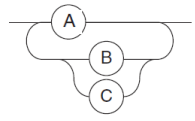
(B) Lexical

(C) Logic

(D) Syntax

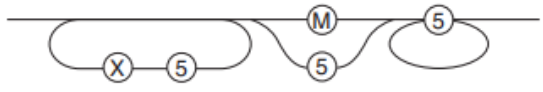
2013 HSC Q15

15 Which of the following correctly matches an EBNF statement with its railroad diagram?

(A)	A {A B}	
(B)	A | B | C [B]	
(C)	{A < B >}	
(D)	A [{B} | {C}]	

2018 HSC Q12

A railroad diagram is shown.



Which EBNF definition is equivalent to the railroad diagram?

A. { 5 X } (M | 5) 5 { 5 }

B. { X 5 } (M | 5) { 5 }

C. { 5 X } (M | 5) { 5 }

D. { X 5 } M | 5 { 5 }

2014 HSC Q27A)

A piece of custom software was used by a business for one year before errors started occurring. The software has not been updated since it was installed.

(a) Outline TWO possible reasons for the errors occurring after one year. **3**

Updates:
New OS or HW
Calculation changes i.e. GST/ Commission
Time:
Testing previously incomplete/ not thorough
Capacity – HDD or Network bandwidth
Business growth – volume testing not completed

Criteria	Marks
• Outlines TWO possible reasons	3
• Outlines ONE possible reason OR • Identifies TWO possible reasons	2
• Identifies a possible reason	1

Answers could include:

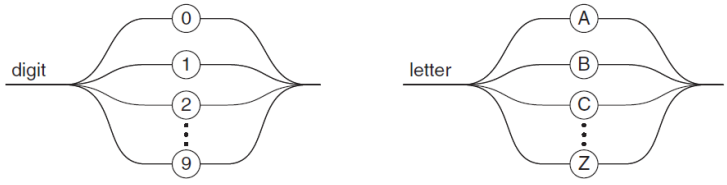
- The software has probably not been thoroughly tested. There could be a pathway in the software that has only recently been followed because its conditions have only recently occurred.
- Some capacity in the software has finally been reached, eg hard disk space or network bandwidth.
- Something may have changed in the environment, for example the operating system, some hardware, or other programs on the system. The new hardware or software may be incompatible with the custom software.
- Changes in the program's calculations due to external factors. For example, if the GST rate was raised from 10% to 15%, and the custom software was not updated, then all GST calculations would be wrong.

2014 HSC Q32

DIM statements are used to declare arrays in a particular programming language, for example: **3**

```
DIM LIST[4] as string
DIM TABLE[3,125] as string
DIM MULTI[4,6,3,12,9] as integer
```

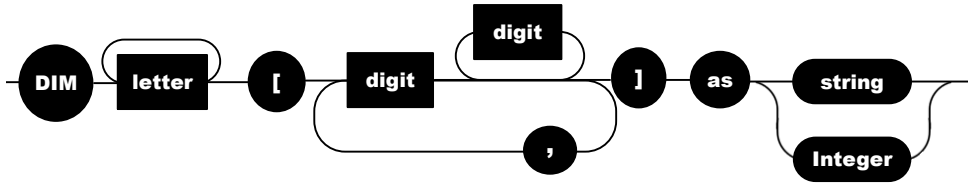
A digit and a letter are defined in this programming language as follows.



Draw a railroad diagram to define a DIM statement in this programming language.

```
DIM LIST[4] as string
DIM TABLE[3,125] as string
DIM MULTI[4,6,3,12,9] as integer
```

If the element is already defined use a ▭ otherwise use a ◯



Criteria	Marks
• Provides a substantially correct railroad diagram incorporating all components	3
• Provides a railroad diagram that illustrates some features of the DIM statement	2
• Shows a basic understanding of railroad diagrams	1

2018 HSC Q27

The syntax rules of the programming language XYZ are described below. **4**

```
Program = START <Identifier> { <Statement> # } END <Identifier>
Statement = <Repetition> | <Selection> | <Sequence>
Repetition = LOOP <Integer> { <Statement> # } ENDLOOP
Selection = IF <Condition> { <Statement> # } OTHERWISE { <Statement> # } ENDIF
Sequence = <Inc> | <Print> | <Assign>
Inc = ++ <Identifier>
Print = Display <Identifier>
Assign = Put < Integer> into <Identifier>
Identifier = <Letter> { <Letter> } <Digit>
Letter = A | B | C | D | E
Digit = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
Integer = <Digit> { <Digit> }
```

The following program is written in XYZ.

```
10      START A
20          Put 20 into F1 #
30          LOOP 5
40              ++ F1 #
50              Print F1 #
60          ENDLOOP
70      END A
```

This program contains some syntax errors.

With reference to the syntax rules provided, explain each of the errors.

Criteria	Marks
• Identifies the syntax errors and explains the illegal syntax	4
• Identifies some syntax errors AND explains some of the illegal syntax	3
• Identifies some syntax errors OR explains the illegal syntax of one error	2
• Shows some understanding of syntax rules	1

Sample answer:

1. The identifier A is invalid in lines 10 and 70, as it should be terminated by an integer (definition for Identifier)
2. The identifier F1 is invalid, as F is not a valid letter (definition for Letter)
3. The ENDLOOP statement should be followed by a #, as each statement (including repetition) is to be followed by a # (definition for Program)
4. Print is not a valid verb. It should be 'Display'.

2018 HSC Q29

The three main steps in the translation process are: **3**

- lexical analysis
- syntactical analysis
- code generation.

With reference to these three steps, describe how the statement

total = number1 + number2

is translated into machine code.

Criteria	Marks
• Describes how the statement is translated with reference to the three steps	3
• Outlines some steps in the translation process	2
• Shows a basic understanding of one of the steps	1

Sample answer:

Lexical analysis: The statement of the source code is scanned and the lexemes Total, =, number1, +, number2 are identified and each allocated a unique token.

Syntactical analysis: This tokenised stream is now scanned against the relevant syntax rules.

Because the statement begins with a variable, it must be an assignment statement, for which a valid rule is:

<Variable> = < variable > < operator> < variable>

It then matches each of the tokens to this structure, with total, number1, number2 being variables, and + being an operator.

Code generation: The machine code statements are generated by converting the tokenised stream to equivalent machine code statements, making use of the available operations in the instruction set of the computer.

2017 HSC Q30

Describe what happens during the fetch–execute cycle. **3**

.....

Criteria	Marks
• Provides a detailed description of the fetch–execute cycle	3
• Provides features of the fetch–execute cycle	2
• Shows some understanding of the fetch–execute cycle	1

Sample answer:

A copy of the next instruction in RAM is placed in the instruction register.
The program counter is incremented by the number of bytes in the instruction.

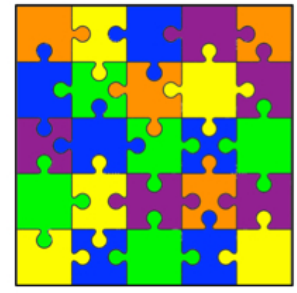
The instruction is decoded and carried out. Results from calculations are stored.

Testing & Evaluating:

- **Where the problem may be, what type of error is occurring, which type of testing is needed and why**
- Module Testing – a test that each module and subroutine functions correctly -- DRIVERS
- Program Testing – a test that the overall program functions correctly as a whole
- System Testing – testing that hardware, software, data, personnel and procedures that form the final system are able to work together efficiently, correctly

Levels of Testing

Module Testing : Test each parts independently and isolated



Program Testing : Make sure that different pieces work together. Test the Interfaces between the different pieces. Interaction with other systems (Hardware, OS, etc.)

System Testing : Test the whole system

Testing & Evaluating:

- Design relevant Test Data
- Live Test Data - to ensure that the testing environment accurately reflects the expected environment in which the new system will operate
  - Large files
  - Mix of transaction types
  - Response times
  - Volume of data
  - Benchmarking
  - effect of the new system on the existing systems in the environment into which it will be installed

Testing & Evaluating:

- Report on testing process
- Relevance - Meets original requirements?
- Quality Assurance - measured against how well the product meets or exceeds users' expectations – You should be able to discuss the 18 elements
- Communication with customer/users
- Post Implementation review – Client sign off

QUALITY ASSURANCE

Clarity	the meeting of a set of standards for the user interface e.g. precise and unambiguous instructions.
Correctness	the consistent production of the correct output for a given set of inputs.
Documentation	Consistent, complete documentation and of a high standard.
Economy	software that is economical both of processing needs (such as main storage needs) and external resources (such as user time).
Efficiency	A measure of how well the program does what it is designed to do
Flexibility	the ability to cope with all the situations found during processing.
Integrity	the ability of a system to withstand any attacks on its security, whether or not these attacks are intentional.
Interoperability	the ability of the software product to communicate with pre-existing software. E.g. Operating System
Maintainability	the ease with which an error in the software can be corrected.
Modifiability	the ease with which the software can be changed to meet new needs or circumstances.
Modularity	being able to replace one part without creating a completely new product.
Portability	the ability of software to be executed with different hardware and software combinations.
Reliability	a measure of the failure rate of software.
Resilience	a measure of the software's ability to recover from an abnormal situation.
Reusability	a quality built into the software at design time when an effort is made to ensure that the components being assembled for the current task can be used in future development activities.
Testability	testability-how easily the software can be tested.
understandability	how well the design of the software is understood.
Usability	a measure of a number of aspects of importance to the user and how the software product meets the specifications of the user.

2013 HSC Q16

16 Which of the following best describes *acceptance testing*?

(A) Testing that the system can accept large volumes of data

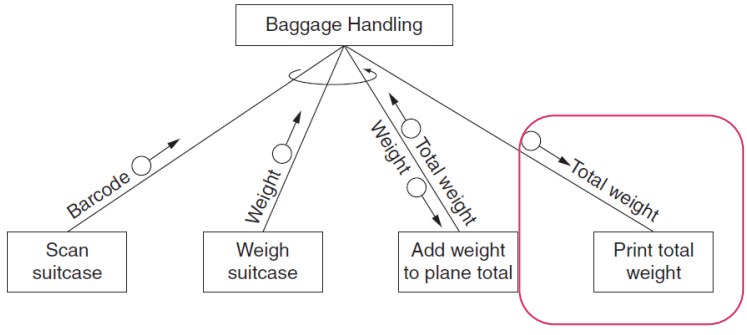
(B) Testing the user-friendliness of the interface by the end users

(C) Testing whether the response times are acceptable to the end users

(D) Testing whether the system is ready to become available to the end users

2013 HSC Q29A)

At an airport the maximum baggage limit per item is 20 kg. The following structure chart represents part of the baggage handling system.



(a) It is found that the Print total weight subroutine is always outputting a value between 0 and 20 kg. Describe strategies for locating the error in the system. **3**

Criteria	Marks
• Describes appropriate strategies	3
• Describes an appropriate strategy	2
• Identifies a feature of testing	1

- It is necessary to separately check each subroutine in the system. The error could be in a single or multiple subroutines.
- **Scan suitcase** can be tested by coding a debugging output statement after the scan is done and then compare the output with the actual barcode number.
- **Weigh suitcase** can be tested by putting a known weight on the scales and then calling the subroutine. A debugging output statement in the calling subroutine, Baggage Handling, could then output the weight. This will check that the scales are correct and that the data is being correctly passed back to the calling subroutine.
- **Add weight to plane total** can be tested by weighing a known weight twice and putting a debugging output statement after the call to Add weight to plane total in the calling subroutine. For example, if the known weight is 10kg, we would expect the output statement to output 10kg and then 20kg. If it outputs 10kg then 10kg it would indicate that the weight is not being added correctly and a desk check should be conducted of Add weight to plane total.
  - Drivers could also be used to pass information to the subroutine and output information to test the calculations.
- **Print total weight** can be tested by using a driver and passing it a known weight, e.g. 1000kg, and then checking that is what is printed.

2014 HSC Q29

A real estate agency uses a system of three programs: **3**

- Sell — list properties, record owner details
- Buy — search for properties, record buyer details
- Rent — calculate rent, record leasing information.

The agency has checked that the leasing information is correct, but tenants have reported that they have been charged too much rent.

Which level(s) of testing (system, program, module) should be used to isolate the error? Justify your answer.

Criteria	Marks
• Recommends appropriate level(s) of testing with justification and reference to the scenario	3
• Recommends a level of testing with appropriate justification	2
• Shows a basic understanding of levels of testing	1

- The error obviously exists in the Rent program. Therefore system testing is not necessary.
- The incorrect calculation is likely a result of incorrect logic as the leasing information is correct. Module testing will need to be done on the Renting module to identify and then fix the logic errors in the code. The module should be tested with appropriate test data and desk checked to ensure that the calculation logic is correct.
- Once corrected and complete the whole Rent program should be tested (program testing) to ensure the data is passing without errors.

2018 HSC Q26

During a one-day sale, an online shopping site became unresponsive when too many people attempted to visit the site at the same time. **3**

Describe the testing techniques that could have been used to prevent this issue from occurring.

Criteria	Marks
• Describes relevant testing techniques that could have helped prevent the issue	3
• Outlines one relevant testing technique that could have helped prevent the issue	2
• Shows some understanding of the problem	1

Sample answer:

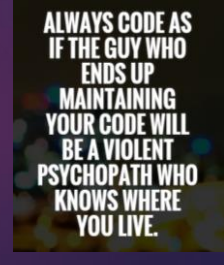
The developer should have predicted the maximum load on the system at peak times, and tested their system with a large volume of data. This should include a mix of transaction types (such as searching, browsing, purchasing items and amending orders) and the use of large files of data for products and orders.

This would have alerted the developer to potential issues with response times and allocated file space before the system went live.

- Some key terms:
- Before, live, response times, peak, volume

Maintenance:

- Modifying code to meet changed requirements
- An ongoing process of correction and refinement to continue to meet the expectations of their users.
- includes the correction of errors (bugs) in the source code & upgrades to enhance the functionality to meet new or changing requirements.
- Documenting changes
  - Comments in source code **(& Intrinsic Documentation)**
  - Updating documentation (hard copy, online Help)
  - Use case tools to monitor changes & versions



2013 HSC Q8

8 Which of the following is NOT a valid reason for maintaining software?

(A) Existing software also has to work on hand-held devices, such as tablets.

(B) Equal opportunity legislation demands greater inclusivity in user interfaces.

(C) There are legislative changes, such as the introduction of the GST, requiring added functionality.

(D) The increased speed of the national broadband network (NBN) allows users to download software more efficiently.

2018 HSC Q4

Which of the following is part of software maintenance?

A. Defragmenting data files on a regular basis

B. Modifying code in response to changed requirements

C. Ensuring that software is compatible with a range of hardware

D. Making sure all personnel are trained in the use of any upgraded software

2014 HSC Q28

The following algorithm is intended to store the results of the times tables of the numbers 1 through to 6 in the array product. The algorithm has logic errors. **3**

```
BEGIN
    row = 1
    WHILE row < 6
        FOR column = 1 to 6
            product (row, row) = column * column
        NEXT column
    END WHILE
END
```

The expected content of the array product is shown below.

1	2	3	4	5	6
2	4	6	8	10	12
3	6	9	12	15	18
4	8	12	16	20	24
5	10	15	20	25	30
6	12	18	24	30	36

Criteria	Marks
• Provides a correct algorithm that fixes the logic errors	3
• Provides a substantially correct algorithm that fixes the logic error(s)	2
• Shows an understanding of a logic error	1

Modify the algorithm so that it operates correctly.

```
BEGIN
    row = 1
    WHILE row ≤ 6
        FOR column = 1 to 6
            product (row, column) = column * row
        NEXT column
        row = row +1
    ENDWHILE
END
```

2017 HSC Q26

The two algorithms shown below both achieve the same thing. **3**

Algorithm 1

```
BEGIN Pay
input Hours
IF Hours > 7 THEN
BaseHours = 7
Overtime = Hours – 7
ELSE
BaseHours = Hours
Overtime = 0
ENDIF
BasePay = BaseHours * 50
OvertimePay = Overtime * 50 * 1.5
TotalPay = BasePay + OvertimePay
Display TotalPay
END
```

Algorithm 2

```
BEGIN Pay

   BaseRate = 50
   PenaltyRate = 1.5
   NormalHours = 7
   Overtime = 0

   input Hours

   IF Hours > NormalHours THEN
      BaseHours = NormalHours
      Overtime = Hours – NormalHours
   ELSE
      BaseHours = Hours
   ENDIF

   BasePay = BaseHours * BaseRate
   OvertimePay = Overtime * BaseRate * PenaltyRate
   TotalPay = BasePay + OvertimePay

   Display TotalPay

END
```

Explain how features of *Algorithm 2* make it easier than *Algorithm 1* to maintain.

- What is the feature
- What is the outcome of it being included
- How does not including these issues affect MAINTENANCE

- Indenting
- Spacing/ White space
- Global Variables initialised

Any Final Questions/ Concerns?

Exam tips and moving up a mark range – By Anil Warrier

Ask the Audience – What is your idea of increasing your marks.

Input questions.

Survey questions

Survey Questions

- Feedback on Survey questions.
- Total of _____surveys were sent out.
- Total of _____completed surveys were received.
- _____ of the people said that they were prepared, which means, they averaged more than 2 in their responses.

Before the exam

- Organise yourself and your study space. Creating a timetable with the amount of work that needs to be done is a good idea.
- Chunk large topics in to smaller topics, so that it does not seem overwhelming.
- Use flowcharts and diagrams. (Double benefit for those in Software design)
- Practise on Old exams.
- Take regular breaks.
- Plan your exam day.
- Look after your health.
- Stay hydrated.
- Sleep well and enough, so that you wake up refreshed the next day.

During the exam.

- Attempt all questions. Do remember you cannot be penalised for a wrong answer. If you answer multiple choice question, rather than leave it empty, there is a ¼ chance that you will get it right.
- Be aware of the limited time that you have available. You have around 30 to 45 seconds(maximum) for every mark which means, you can spend around 1 minute to 1.5 minute for a 2 mark question.
- Be very aware of the time. If in doubt, please read point number 2 again.
- If you doubt that you cannot answer any question to the best of your ability, please move on.
- Do remember to return back to the question.
- Remember to leave some time to check that you have answered all questions and all sections correctly. You can check this intermittently as you finish a section.

During the exam-(Continued)

- You may choose to do easier questions first. This may help increase your confidence.
- Do remember to number your questions and sections correctly.
- Do not stress, if you do not know the answer to a particular question. Try and use the time towards the end to review it again.

Actual Example(From Board of studies website)

- Example-22b-Exam paper 2018
- :10 BEGIN SubString(name, start, finish)
- 20 set len to the number of characters in name
- 30 set result to “ ”
- 40 WHILE start ≤ finish
- 50 append start the character of name to result
- 60 add 1 to start
- 70 IF start > len THEN
- 80 set start to 1
- 90 ENDIF
- 100 ENDWHILE
- 110 return result
- 120 END SubString
- This algorithm contains a logic error and does not always produce the expected result.

- Explain how breakpoints and single line stepping can be used to locate and confirm the source of the error.
- It is better to answer something rather than nothing as shown below.
- **Sample answer:**
- Breakpoints are useful when infinite loops occur. The execution of a loop, such as in lines 40–100, can be interrupted so that variables can be inspected. This would show that the value of start never reaches the value of finish.
- Single line stepping can show how the contents of variables change with each line of code, demonstrating the unintended values of start.

During the exam(Continued)

- **Question 22 (b) Criteria** **Marks**

- Explains the use of breakpoints and single line stepping for scenario:3
- Explains the use of breakpoints OR single line stepping :2
- Shows some understanding of breakpoints OR single line stepping :1

Programming Paradigms

This is one of the two option topics in your SDD course.

Definition of Paradigm

- Does anyone know of the definition of paradigm?
- Please provide your answers on chat(2 minutes)

Programming Paradigms

- Programming languages are represented as a list commencing at first generation through to fifth generation.
- This list presents programming languages in order of their emergence and increasing user friendliness.

Programming paradigm

- It is a style or way of programming.
- It is a set of rules
- It provides a structure to solve the same problem from different angles.
- Paradigm is description of a new association so it can be formalised and reused.

Development of the different paradigms

- Initial Languages(imperative languages) sequencing, decisions and repetition as their main problem solving methods. Data and processing are separated.
- Computers were initially designed to solve mathematical and arithmetical problems.
- However, many real life problems do not have definite answers or they have a variety of answers.
- Examples: Doctor's diagnosis based on science and experience. Driving a car is more than just operating the controls. Communication between people of different cultures. Tone, Volume and inflections of voice have different meaning

- What does this mean?

- It does mean that the task of software developers will be simplified.

Programming Paradigms

- It could be preferable if code could be used to solve related problems without the need to be rewritten. This reusability should be an integral part of the paradigm in much the same way that our brain uses past experience to solve new related problems.
- Is there a paradigm that encourages the use of code?

- Are you aware of any industries that reuses components.
- Can you give some examples?

- Examples: 12 volt bulb – In a torch, airplane, car or a boat

Programming Paradigms

- As software developers, we need to be able to chose a paradigm most suited to the current problem.

Logic Paradigm

- Logic paradigm uses facts and rules as its basic building blocks. It is used to develop programs in the area of Artificial Intelligence.
- Example:
- Rule 1: If a plane has a jet engine and a single seat, then it is a jet fighter.
- Rule 2: If a plane has a jet engine and a pressurised cabin, then it can fly above 15000 feet
- Rule 3: If a plane has fixed windows, then it has a pressurised cabin.
- Rule 4: If a plane can fly above 15000 feet, then it must be air-conditioned.

- Prove: A jet with fixed windows must be air-conditioned.
- Can you try to prove the above using the above rules?

- Backward chaining-Assume the theory is true and ask questions.
- A jet with fixed windows must be air-conditioned.
- Search for the rule – Plane must be air-conditioned
- Rule 4 tells us that if a plane must be air-conditioned, they can fly above 15000 feet.
- Rule 2 tells us that if a plane has a jet engine and a pressurised cabin then it can fly above15000 feet.
- Rule 3 further states that any plane that has fixed windows must have a pressurised cabin.

- Forward chaining-Start from the beginning of the facts and rules.
- Use our knowledge that our Jet has fixed windows
- Rule 3 means that our plane must also be pressurised, because, it has fixed windows.
- Rule 2 says that our plane can fly above 15000 feet.
- As a consequence Rule 4 becomes relevant and our plane must be air conditioned.

- Thinking points
- Examine the following statements in Prolog- Leading programming language in Artificial Intelligence.
- eat(lion,dog).
- eat(dog,cat).
- eat(cat,bird).
- eat(bird,spider).
- eat(spider,fly).

- Can you examine the answers to the following?
- caneat(lion,fly) = yes
- caneat(bird,dog)=no

- How would you reach the conclusion to the statements above using the explanation above?

Programming Paradigm

- Object Oriented Paradigm
- Reusability of code
- Uses Objects which contain attributes and methods.
- Attributes – Stuff that an object knows and remembers
- Methods-Attributes can only be accessed and altered by the object's methods.
- For example: Prolog has versions that include OO functionality.

- Example: Adventure game
- Object-Mr-Nice- Attributes-Timid, Not very agile, very intelligent, Quite healthy
- Method-Move around, Get treasure, Fight
- Values of Mr. Nice's attributes can only change as a result of one of his methods altering the attribute. The concept is known as encapsulation. Mr. Nice's attributes are private and contained within him. Encapsulation means, that objects control their own private attributes using their own methods. This helps in creation of robust and reusable classes of objects.

- If each attribute can be represented using an integer from 0 to 10, where 0 is low and 10 is high, we can create a class of Mr. Goodie. Mr Nice may be a certain Mr. Goodie with certain attributes, which can change. For example: Mr Goodie can outwit Mr Baddie and his intelligence can rise from 5 to 9.
- Declaring a class is in many ways similar to declaring a record as a new data type in a procedural language.
- In OOP, these definitions are called classes and a particular instance of class is called an object.

- For example, we could declare our goodie class with the following code

```
public class Goodie{
        private int personality;
        private int agility;
        private int intelligence;
        private int health;
        public void move(){
}
public void fight(){
}
public void get treasure(){
}
```

- Or we could create a particular instance of the object as below
- Goodie MrNice;
- MrNice=new Goodie();
- Goodie MissLovely=new Goodie();
- GoodieMrWild=new Goodie();

- Can you create a baddie class declaration with the attributes of agility, health, aggression and bad type? (5 minutes)
- Is there a similarity between the goodie class and baddie class?

- What would happen, if we could create a super class called character, which contain all the common attributes and methods used in both the baddie and goodie characters?
- The subclass goodie and baddie characters can then said to be inheriting the characteristics of the super class.
- In some OOP languages, it is possible for a class to inherit the characteristics of a number of super classes. This is termed as multiple inheritance.
- Additionally, if we want a method to do something, like, all our goodies with both agility and health set at 5, we need function definitions called constructor methods.

Programming Paradigms - Conclusion

- Programming paradigm or a style of programming was developed as a need to define programming languages in a different way other than just generational languages.
- It was developed since, all problems may not have one standard solution or may have more than one solution.
- It was felt that reusability of code is important.

References

- Davis, Sam and Fendall,Janine (2012). *Software Design and Development*. Second Edition. Parramatta Education Centre, (361-411)

- Questions
- Revision worksheet

Question 1.

Imperative languages use control structures of sequence, selection and repetition together with local and global variables. Explain how logic languages such as Prolog achieve the same outcomes without all of the control structures, local and global variables.

Question 2.

(a) Describe ONE significant reason for the development of the *object oriented* paradigm.

(b) Explain the use of *encapsulation* within the *object oriented* paradigm and provide an example.

(c) Discuss features of the *object oriented* paradigm that affect the programmers' productivity with particular reference to speed of code generation.

Question 3.

A scientist is trying to classify various rock materials. Many experiments have been performed to determine the features and characteristics of each sample. Features and characteristics include colour, atomic weight and structure, abundant elements plus location found.

With all of this information the scientist has discovered that it possible to classify any rock material into one of the categories.

The scientist wants his assistants to be able to carry out the classification process without him present and therefore requires a system to be developed which accepts the necessary information about the sample and classifies it.

Recommend the most appropriate paradigm for the solution of this problem. Explain why other paradigms would be less appropriate.

Suggested solutions

Question 1.

Imperative paradigms use the repetition control structure to repeat statements whereas the logical paradigm uses recursion. The predicate is called initially and for each subsequent call the argument is modified until it reaches the termination condition. Logical languages have a stopping (or terminating) condition in their code. Selection in Prolog is implemented by having different versions of each predicate which will match with different facts or rules.

With regards to variables, imperative languages allow the use of global variables resulting in a situation where a value can be seen and hence modified anywhere within the program. With logical languages values can only be seen in a predicate if they are passed as parameters. Prolog variables are used to match or bind with known facts during the goal seeking process. This is different to imperative language variables which are assigned values specifically by the programmer or by the detailed logic of the solution.

Question 2.

(a) One significant reason for the development of OOP was that it simplifies the reuse of code. This reduces repetitive programming tasks and is achieved through the use of classes of objects and their associated data and methods. Once a class has been defined it can be reused in any programming project without the need to redefine the data structures and operations.

(b) The principle of encapsulation is essential to the OOP. It establishes a firewall between the user of an object and the code implementing it, thus achieving information hiding. In particular the user does not need to know how the data or attributes of an object are defined. If we had a Stack class with pushing and popping operations the stack could be implemented as an array, a list or whatever, since the representation of the Stack is unknown outside of the object. The only way to interface with the object is via the methods.

(c) OOP increases programmers' productivity because of the inheritance feature. Once a class has been defined and tested it can be used by the programmer. If the programmer wishes to extend the class into a sub class they do not need to rewrite, and hence test, all of the methods needed to operate an object of the sub class but can call the methods from its super or parent class, this means only additional methods need to be defined and tested which increases code generation time.

Question 3.

The logic paradigm would be the most appropriate as the scientist has come up with some rules on how to classify the rock material. These rules as well as all the known facts about existing rocks can be entered into a Prolog system or perhaps into an expert system shell. The assistant will then be able to enter the facts about the current sample to be classified and the logic paradigm (or expert system) will then use its rules to determine the correct classification.

The imperative and OOP paradigms are unsuitable. Both these paradigms require algorithms and then code which explicitly describes how to solve the classification problem. A different algorithm (and hence subprogram) would be needed to solve each different classification method. Therefore, any OOP or imperative program written to solve this problem will be long and complicated with lots of cases listed. Subsequently, it can be seen that the most appropriate paradigm is logic.

Notes

- Questions which ask about the most appropriate paradigm do not necessarily have a single correct answer. It is often possible to describe a suitable method of implementing a solution using various different paradigms.
- The logic and object oriented paradigms can be used together as the concepts involved are very different and meet very different purposes. There are object oriented versions of Prolog which attempt to combine the advantages of both logic and object oriented programming

Software Design and Development

Option 2: The interrelationship between software and hardware

AURORA COLLEGE

Overview

- Representation of data
  - ASCII versus Unicode
  - Hexadecimal, binary, decimal
  - 1's and 2's complement
  - Floating point representation
  - Binary arithmetic
- Circuits
  - Logic gates: AND, OR, NOT, NAND, NOR, XOR
  - Truth tables
  - Boolean algebra
  - Circuit design
  - Specialty circuits: half adder, full adder, flip-flops
- Programming of hardware devices
  - Data stream format and interpretation
  - Using input from sensors and other devices

AURORA COLLEGE

Representation of data

AURORA COLLEGE

Representation of data

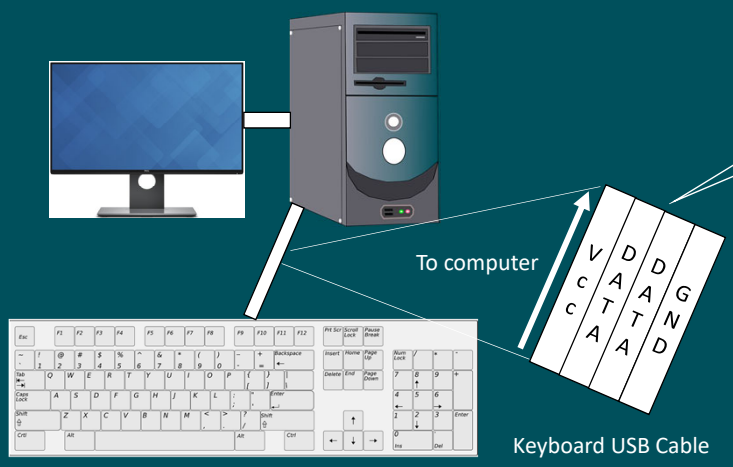
- ASCII: American Standard Code for Information Interchange
- Unicode
- Binary, decimal and hexadecimal numbering system
- Representing negatives in binary: 1's and 2's complement
- Floating point representation
- Binary arithmetic
  - Addition
  - Subtraction
  - Multiplication (shift left or up)
  - Subtraction (shift right or down)

AURORA COLLEGE

ASCII and Unicode

- All data input from keyboards must be coded in a way a computer understands.
  - ASCII: American Standard Code for Information Interchange – 7 bits with 1 bit parity check
  - Unicode: Universal code – up to 32 bits. Represents most of the world's writing systems.

ASCII and Unicode



Either one or both of these data channels can transmit the 0s and 1s for EACH key pressed on a keyboard.

For example:
'a'
– ASCII: 61_{16} or $110\ 0001_2$
– Unicode: 61_{16} or $0110\ 0001_2$

Copyright symbol
– ASCII: Does not exist
– Unicode: $A9_{16}$ or $1010\ 1001_2$

Decimal

	10^4	10^3	10^2	10^1	10^0
Value	**10 000**	**1 000**	**100**	**10**	**1**
9					9
0				1	0
32				3	2
99 999	9	9	9	9	9
1024		1	0	2	4

- Each column from right to left is an increasing power of ten.
- Up to 10 digits (0 – 9) can be placed in each column before the next one to the left has to be used.
- Most common numbering system in the world.

AURORA COLLEGE

Binary

	2^3	2^2	2^1	2^0
Value	**8**	**4**	**2**	**1**
0001	0	0	0	1
0010	0	0	1	0
0011	0	0	1	1
1100	1	1	0	0
1110	1	1	1	0

- Each column from right to left is an increasing power of two.
- Up to 2 digits (0 – 1) can be placed in each column before the next one to the left has to be used.
- Most common numbering system for computers in the world.
- Very difficult for humans to read and understand.
- Trailing zeros are nearly always written unlike decimal numbers. Normally in groups of 4.

AURORA COLLEGE

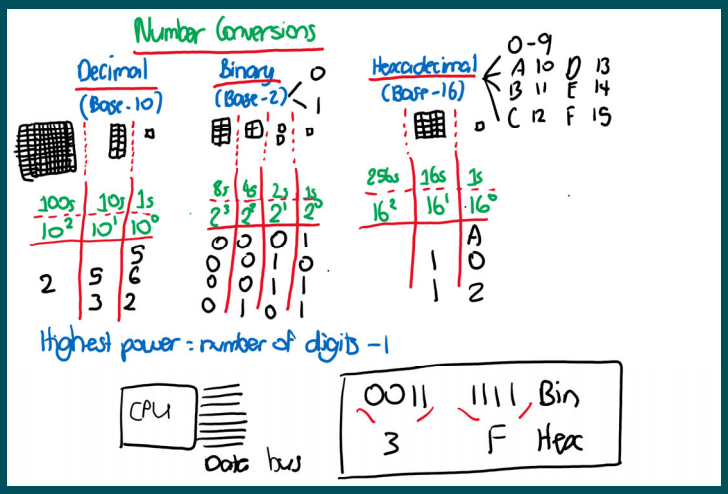
Hexadecimal

	16^3	16^2	16^1	16^0
Value	**4098**	**256**	**16**	**1**
10			1	0
F				F
16			1	6
100		1	0	0
5A			5	A

- Each column from right to left is an increasing power of sixteen.
- Up to 16 digits (0 – F) can be placed in each column before the next one to the left has to be used.
- Used as a go-between for people to read binary numbers. One column in hexadecimal corresponds to 4 columns in binary. This is one of the main reasons for its use – convenience.
- Usually written in groups of two or four. Trailing zeros may or may not be written, depending on the context.

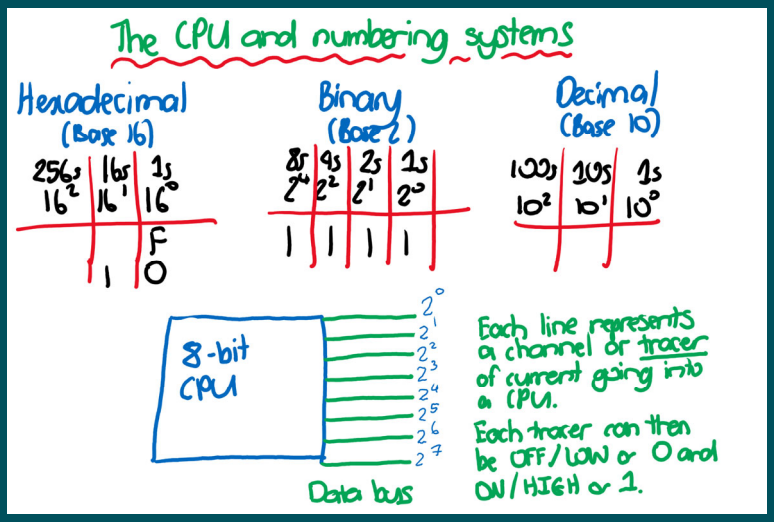
AURORA COLLEGE

Binary – Hexadecimal Relationship



AURORA COLLEGE

Why binary?



Representing negatives in binary

Representation method	Result			
Example: +7	+ = 0		- = 1	
	Sign bit	2^2	2^1	2^0
Signed magnitude	0	1	1	1
1's complement	0	0	0	0
				+ 1
2's complement	0	0	0	1

Flip these bits

Representation method	Result			
Example: -7	+ = 0		- = 1	
	Sign bit	2^2	2^1	2^0
Signed magnitude	1	1	1	1
1's complement	1	0	0	0
				+ 1
2's complement	1	0	0	1

Flip these bits

Floating point representation

This example is for IEEE754 Single Precision (32-bit) standard.

- 3375.00 can be written in Scientific Notation as: 3.375×10^3.
  This can also be: 3.375 E+3.
  In binary this is: 11.011_2
  We thus should have: $1.1011_2 \times 2^1$ in binary Scientific Notation.
- We have a positive, so the sign bit is 0.
- Our exponent is +1, so 127 + 1 = 128 = $1000\ 0000_2$
- We ignore the first 1 before the fraction point, so we store 1011 as shown below.

How?
1. Separate the real and fractional components. Convert the real part (the 3) into binary as normal.
2. Multiply the fractional part by 2, taking note of the whole parts on each iteration, i.e.:

0.375 x 2 = 0 + 0.75
0.75 x 2 = 1 + 0.5
0.5 x 2 = 1 + 0 (stop here)

Now read down: **$0.375_{10} = 0.011_2$**

Sign (1 bit)	Exponent (8 bits)	Mantissa (23 bits)
+ / -	> 127: +ve exponents 127: 0 exponent < 127: -ve exponents	Bits to represent the rest of the numbers.
0	1000 0000	1011 0000 0000 0000 0000 000

AURORA COLLEGE

Floating point representation

This example is for IEEE754 Single Precision (32-bit) standard.

- 0.250 can be written in Scientific Notation as: 2.25×10^{-1}. This can also be: 2.250 E-1.
  In binary this is: 10.01_2
  We thus should have: $1.001_2 \times 2^{-1}$ in binary Scientific Notation.
- We have a positive, so the sign bit is 0.
- Our exponent is -1, so 127 + (-1) = 126 = $0111\ 1110_2$
- We ignore the first 1 before the fraction point, so we store 01 as shown below.

How?
1. Separate the real and fractional components. Convert the real part (the 2) into binary as normal.
2. Multiply the fractional part by 2, taking note of the whole parts on each iteration, i.e.:

0.25 x 2 = 0 + 0.5
0.5 x 2 = 1 + 0 (stop here)

Now read down: **$0.25_{10} = 0.01_2$**

Sign (1 bit)	Exponent (8 bits)	Mantissa (23 bits)
+ / -	> 127: +ve exponents 127: 0 exponent < 127: -ve exponents	Bits to represent the rest of the numbers.
0	0111 1110	0100 0000 0000 0000 0000 000

AURORA COLLEGE

Binary arithmetic: Signed magnitude

Ignore last carry bits

Example 1

Carry	1		1			Decimal
	0	1	0	1		5
+	1	1	0	1	+	(-5)
1	0	0	1	0		2

Example 2

Carry	1	1				Decimal
	1	1	1	1		-7
+	0	0	1	0	+	2
1	0	0	0	1		1

These answers are clearly **not** correct!

Sign	2^2	2^1	2^0	Decimal
Value	**4**	**2**	**1**	
1	1	1	1	-7
1	1	1	0	-6
1	1	0	1	-5
1	1	0	0	-4
1	0	1	1	-3
1	0	1	0	-2
1	0	0	1	-1
1	**0**	**0**	**0**	**-0**
0	**0**	**0**	**0**	**0**
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7

RORA COLLEGE

Binary arithmetic: 1's complement

Ignore last carry bits

Example 1

Carry						Decimal
	0	1	0	1		5
+	1	0	1	0	+	(-5)
1	1	1	1	1		-0

Example 2

Carry						Decimal
	1	0	0	0		-7
+	0	0	1	0	+	2
1	1	0	1	0		-5

We're getting closer …

Sign	2^2	2^1	2^0	Decimal
Value	**4**	**2**	**1**	
1	0	0	0	-7
1	0	0	1	-6
1	0	1	0	-5
1	0	1	1	-4
1	1	0	0	-3
1	1	0	1	-2
1	1	1	0	-1
1	**1**	**1**	**1**	**-0**
0	**0**	**0**	**0**	**0**
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7

RORA COLLEGE

Binary arithmetic: 2's complement

Ignore last carry bits

Example 1

Carry	1	1	1			Decimal
	0	1	0	1		5
+	1	0	1	1	+	(-5)
1	0	0	0	0		0

Example 2

Carry						Decimal
	1	0	0	1		-7
+	0	0	1	0	+	2
1	1	0	1	1		-5

Right every time!

Sign	2^2	2^1	2^0	Decimal
Value	**4**	**2**	**1**	
1	**1**	**1**	**1**	**-8**
1	0	0	1	-7
1	0	1	0	-6
1	0	1	1	-5
1	1	0	0	-4
1	1	0	1	-3
1	1	1	0	-2
1	1	1	1	-1
0	**0**	**0**	**0**	**0**
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7

AURORA COLLEGE

Binary multiplication (shift and add)

15 X 12 = ?
M X Q = C

Step	C	Q	M	Operation
0	0000 0000	1100	0000 1111	Initialisation
1	0000 0000	1100 0110 →	 0001 1110 ←	Right bit of Q is 0: Shift Q right Shift M left
2	0000 0000	0110 0011 →	 0011 1100 ←	Right bit of Q is 0: Shift Q right Shift M left
3	0011 1100	0011 0001 →	0011 1100 0111 1000 ←	Right bit of Q is 1: Add C from Step 2 to M Shift Q right Shift M left
4	1011 0100 ↑ This is the answer.	0001 0000 →	0111 1000 1111 0000 ←	Right bit of Q is 1: Add C from step 3 to M Shift Q right Shift M left

You only have 4 bits, so you only need to run this algorithm four times.

AURORA COLLEGE

Binary division (shift and subtract)

12 ÷ 3 = ?
R ÷ D = Q

Num1 = 12 (1100_2)
Num2 = 3 (0011_2)

Step	Q	R	D	Operation
0			0000 0011	Initialisation
1			0000 0110	Left hand bit of D is 0: Shift D left.
2			0000 1100	Left hand bit of D is 0: Shift D left.
3			0001 1000	Left hand bit of D is 0: Shift D left.
4			0011 0000	Left hand bit of D is 0: Shift D left.
5			0110 0000	Left hand bit of D is 0: Shift D left.
6	0000 0000	0000 1100	1100 0000	Left hand bit of D is 1: End shift operations. Set R to Num1, which is 12_{10} (1100_2) from above. Set Q to 0.
7	0000 0000	0000 1100	1100 0000 1100 0000 0110 0000	As D ≥ Num2: Shift Q left. As NOT R ≥ D: Shift D right.

Binary division (shift and subtract)

12 ÷ 3 = ?
R ÷ D = Q
Num1 = 12 (1100_2)
Num2 = 3 (0011_2)

Step	Q	R	D	Operation
8	0000 0000	0000 1100	0110 0000 0110 0000 0011 0000	As D ≥ Num2: Shift Q left. As NOT R ≥ D: Shift D right.
9	0000 0000	0000 1100	0011 0000 0011 0000 0001 1000	As D ≥ Num2: Shift Q left. As NOT R ≥ D: Shift D right.
10	0000 0000	0000 1100	0001 1000 0001 1000 0000 1100	As D ≥ Num2: Shift Q left. As NOT R ≥ D: Shift D right.
11	0000 0000 0000 0001	0000 1100 0000 0000	0000 1100 0000 1100	As D ≥ Num2: Shift Q left. As R ≥ D: Set R to R - D. Set Q to Q + 1.

Binary division (shift and subtract)

12 ÷ 3 = ?
R ÷ D = Q
Num1 = 12 (1100_2)
Num2 = 3 (0011_2)

Step	Q	R	D	Operation
12			0000 0110	Shift D right.
13	0000 0010	0000 0000	0000 0110 0000 0110 0000 0011	As D ≥ Num2: Shift Q from Step 11 left. As NOT R ≥ D: Shift D right.
14	0000 0100	0000 0000	0000 0011 0000 0011 0000 0001	As D ≥ Num2: Shift Q from Step 13 left. As NOT R ≥ D: Shift D right.
15			0000 0001	As NOT D ≥ Num2: Halt.

The answer $0100_2 = 4_{10}$ The remainder is 0

Circuits

Circuits

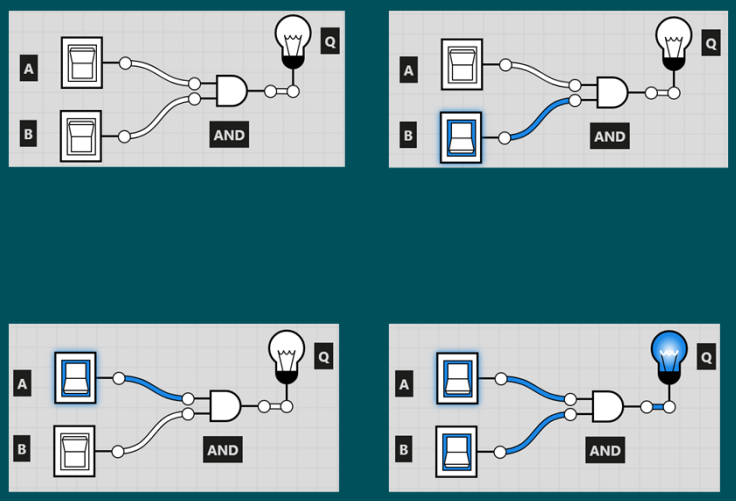
- Logic gates and truth tables: AND, OR, NOT, NAND, NOR, XOR
- Boolean algebra
- Circuit design
- Specialty circuits: half adder, full adder, flip-flops

AND

A	B	Q
Off	Off	Off
On	Off	Off
Off	On	Off
On	On	On

Truth table for AND.

On/Off can be replaced with High/Low if these are used within a real circuit.

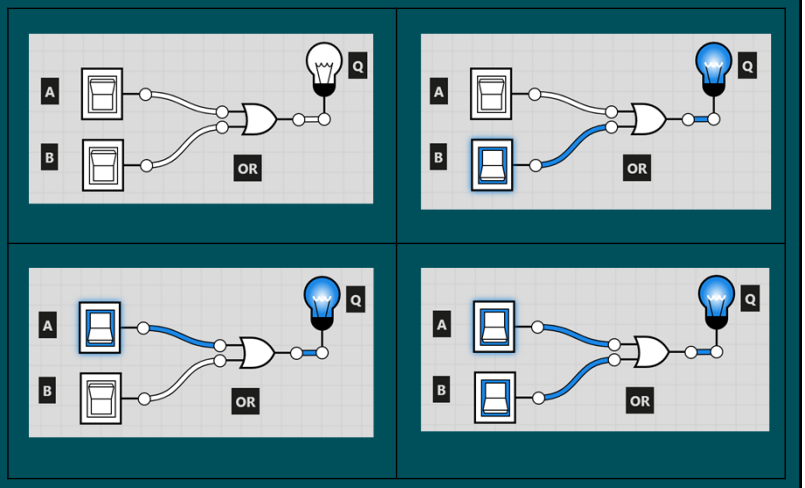


OR

A	B	Q
Off	Off	Off
On	Off	On
Off	On	On
On	On	On

Truth table for OR.

On/Off can be replaced with High/Low if these are used within a real circuit.



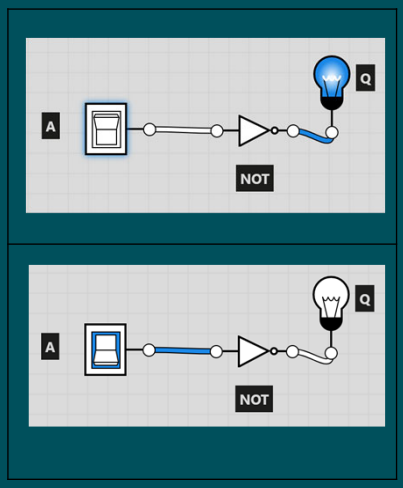
AURORA COLLEGE

NOT

A	Q
Off	On
On	Off

Truth table for NOT.

On/Off can be replaced with High/Low if these are used within a real circuit.



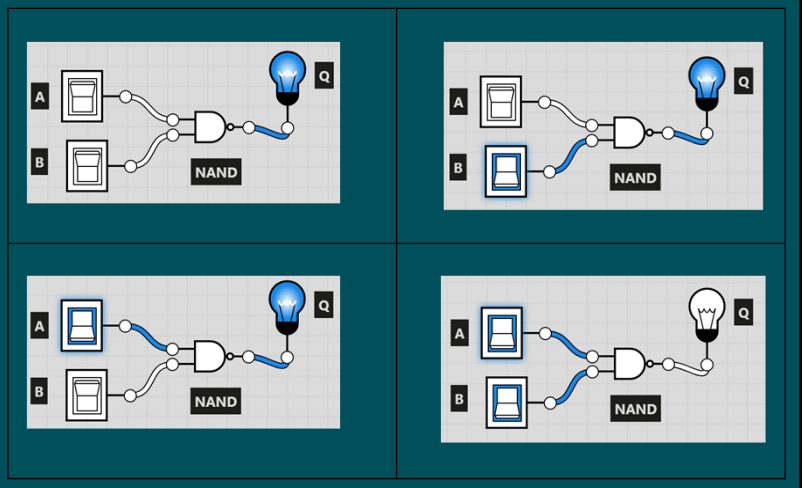
AURORA COLLEGE

NAND

A	B	Q
Off	Off	On
On	Off	On
Off	On	On
On	On	Off

Truth table for NAND.

On/Off can be replaced with High/Low if these are used within a real circuit.



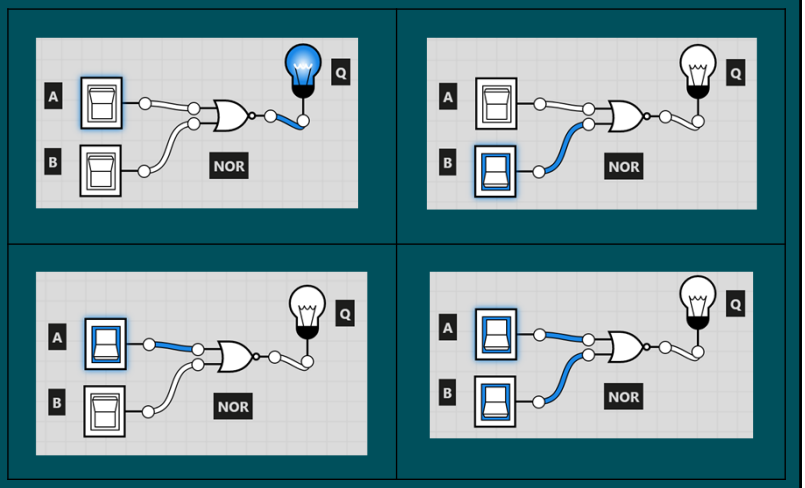
AURORA COLLEGE

NOR

A	B	Q
Off	Off	On
On	Off	Off
Off	On	Off
On	On	Off

Truth table for NOR.

On/Off can be replaced with High/Low if these are used within a real circuit.



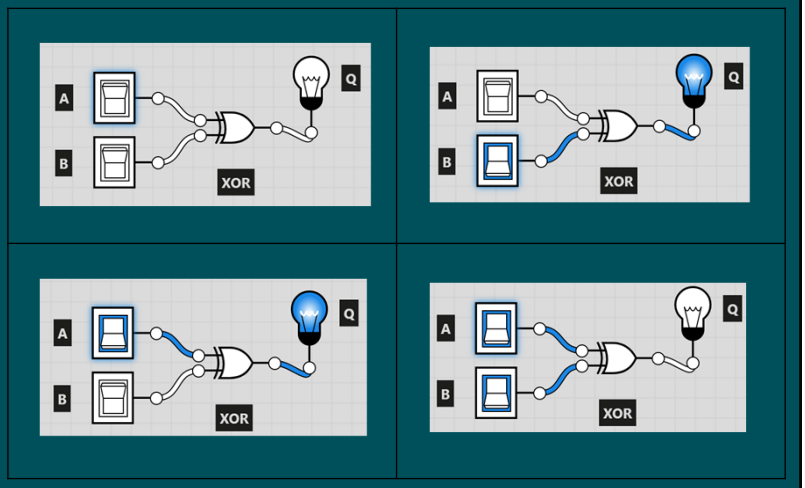
AURORA COLLEGE

XOR

A	B	Q
Off	Off	Off
On	Off	On
Off	On	On
On	On	Off

Truth table for XOR.

On/Off can be replaced with High/Low if these are used within a real circuit.



AURORA COLLEGE

Truth tables

A	B	Q
0	0	0
1	0	0
0	1	0
1	1	1

AND

A	Q
0	1
1	0

NOT

A	B	Q
0	0	1
1	0	0
0	1	0
1	1	0

NOR

A	B	Q
0	0	0
1	0	0
0	1	0
1	1	1

OR

A	B	Q
0	0	1
1	0	1
0	1	1
1	1	0

NAND

A	B	Q
0	0	0
1	0	1
0	1	1
1	1	0

XOR

AURORA COLLEGE

Boolean algebra

- Created by George Boole, the father of modern logic.
- Developed around the same time as Charles Babbage's Analytical Engine, the progenitor to all modern computers.
- Variables can either be 0 (false/off/low) or 1 (true/on/high).
- Four basic logical operations:
  - AND: $A \cdot B$ -> AB
  - OR: $A + B$
  - NOT: $\bar{A}$
  - XOR: $A \oplus B$

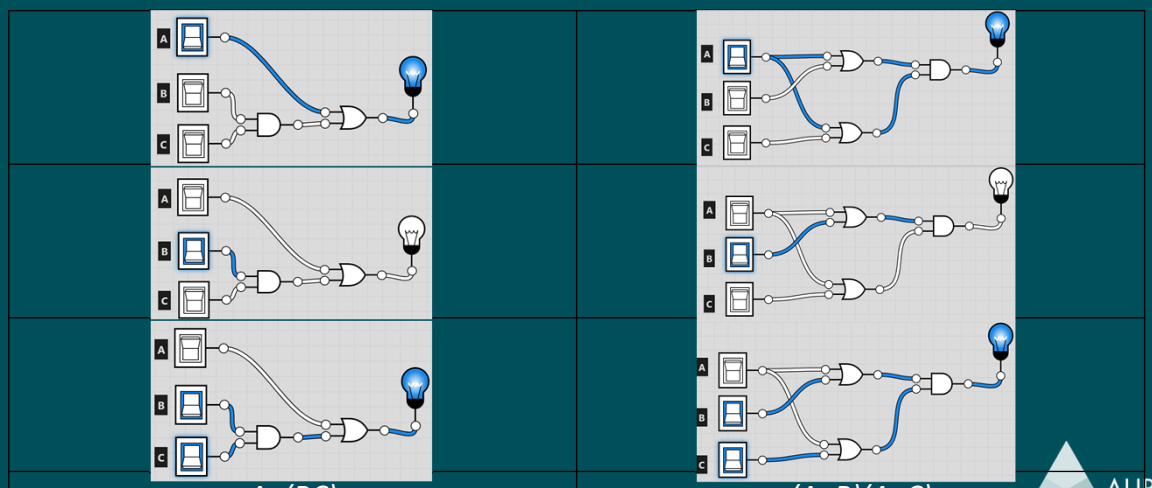
Why Boolean algebra?

- You can use it to design circuits mathematically, then draw them out, then create truth tables to see if it will actually work before doing it in an actual circuit.
- Many of the rules of regular algebra apply to Boolean algebra except for the following:

Boolean algebra rules	De Morgan's Theorem
A+(BC)=(A+B)(A+C)	$(\bar{A} + \bar{B})= \bar{A}\bar{B}$
A+A=A	$(\bar{A}\bar{B})= \bar{A} + \bar{B}$
AA=A	
1+A=1	
$\bar{A}$+A=1	
$\bar{A}$ ·A=0	
A+ $\bar{A}$B=A+B	
A($\bar{A}$+B)=AB	

Why Boolean algebra?

- Let's try: *A+(BC)=(A+B)(A+C)*



Why Boolean algebra?

- You need to be able to interpret Boolean algebraic expressions and turn them into real circuits.
- You need to be able to simplify them to make circuits more efficient using the aforementioned Boolean algebra rules and De Morgan's theorem.
- Two strategies to get a Boolean equation from any truth table:
  1. Sum of Products (SoP): uses all the rows of a truth table with a 1 output
  2. Product of Sums (PoS): uses all the rows of a truth table with a 0 output

You normally use the strategy with the least amount of rows.

Boolean algebra worked example

A	B	Q
0	0	1
0	1	1
1	0	0
1	1	0

Given we have an equal amount of 0s and 1s, it doesn't matter whether we use SoP or PoS. So, using SoP, we only consider the first two lines as shown.

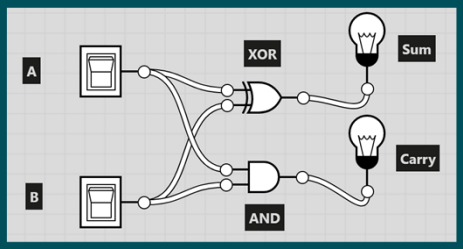
The first row gives us: A = 0, B = 0, therefore $\bar{A}\bar{B}$
The second row gives us: A = 0, B = 1, therefore $\bar{A}B$
The sum of these products, as we are using SoP is:

$$Q = \bar{A}\bar{B} + \bar{A}B$$
$$= \bar{A}(\bar{B} + B)$$
$$= \bar{A}(1)$$
$$= \bar{A}$$

In other words, this truth table is a NOT gate.

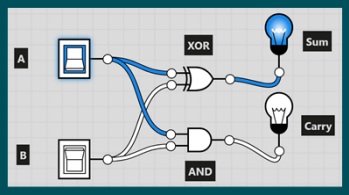
Specialty circuit: half-adder

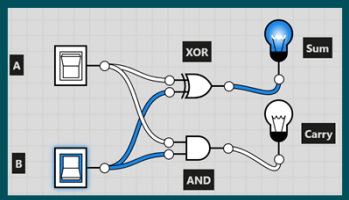
Input		Output	
A	B	Carry	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

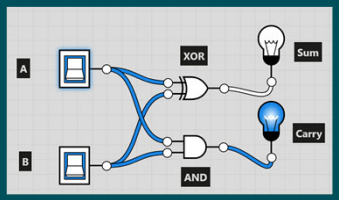


Analysing the behaviour of Carry and Sum, we get:
Carry = $A \cdot B$ -> AND
Sum = $A \oplus B$ -> XOR

Given below is the half-adder circuit.

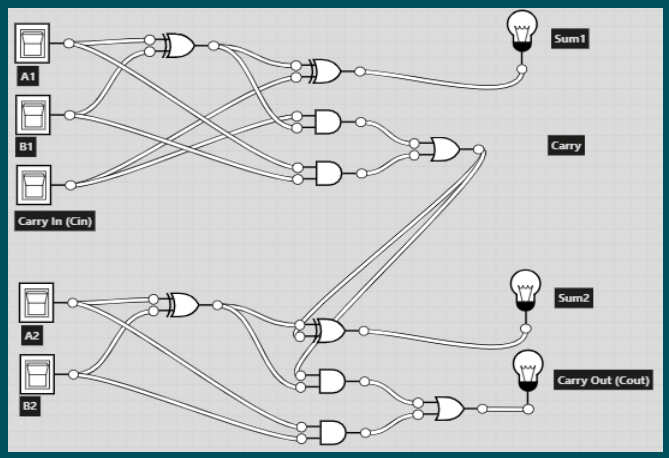






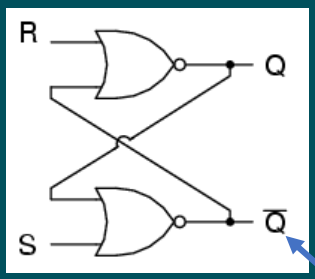
Specialty circuit: full adder

- The main difference is that the carry from the first set of digits gets carried down to the next XOR gate i.e. the first carry is input into the second adder.



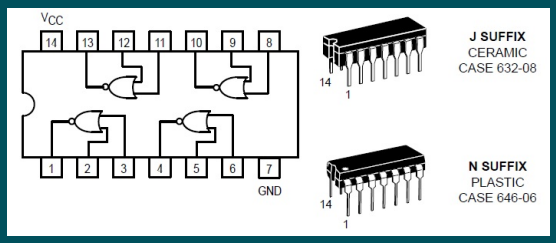
Specialty circuit: flip-flop – S/R Latch

- These types of circuits essentially keep current going to a particular component in a steady state. It is effectively how memory works on a computer.



S/R Latch

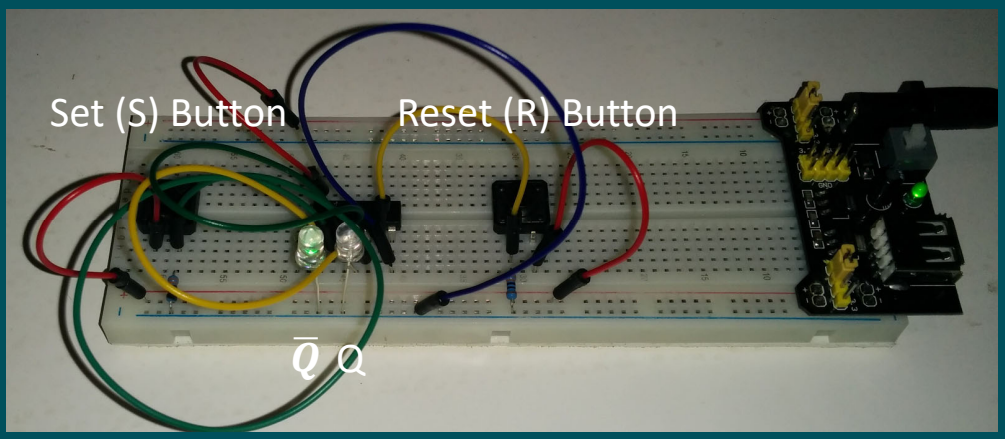
This is also known as **NOT Q**



NOR Chip pin layout

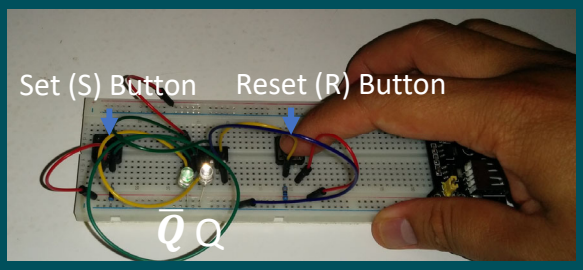
Specialty circuit: flip-flop – S/R Latch

Default state

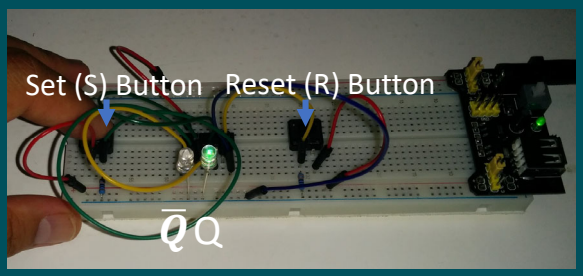


Note: Q is the actual bit in memory which we would want to read. NOT Q is normally a flag telling us that this particular part of memory is either being updated or is empty.

Specialty circuit: flip-flop – S/R Latch



Before R button press



Before S button press

Programming of hardware devices

AURORA COLLEGE

Programming of hardware devices

- Data stream format and interpretation i.e. data packets
- Using input from sensors and other devices i.e. control systems

AURORA COLLEGE

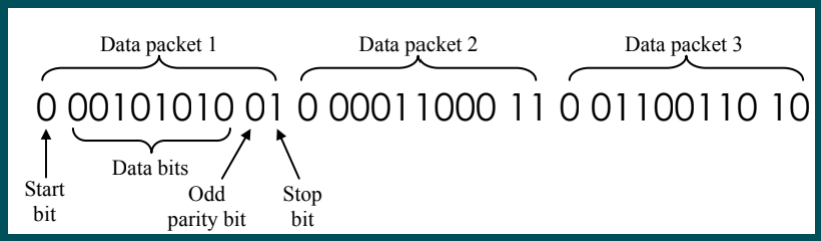
Data packets

- Take a look at this data stream: 00010101001000011000110011001101 0
- What is it?
  - What does it do?
  - Where does it begin?
  - Where does it end?
  - Are there any logical divisions in this data stream? If so, how? Why?
- How do you think you could interpret this data stream?

Data packets

- Data packets are logical organisational units so that computers can interpret the data stream coming in.
- Data packets vary in size, format and purpose but they consist of the same basic structure:
  - **Header** – usually tells the computer where the data is supposed to go
  - **Body** or **message** – tells the computer important data or information for it to do something e.g. X and Y positions of a mouse cursor, characters in an e-mail etc.
  - **Footer** or **tail** or **trailer** – usually gives information from where the data packet came or data it needs to check whether the data packet has been received accurately.

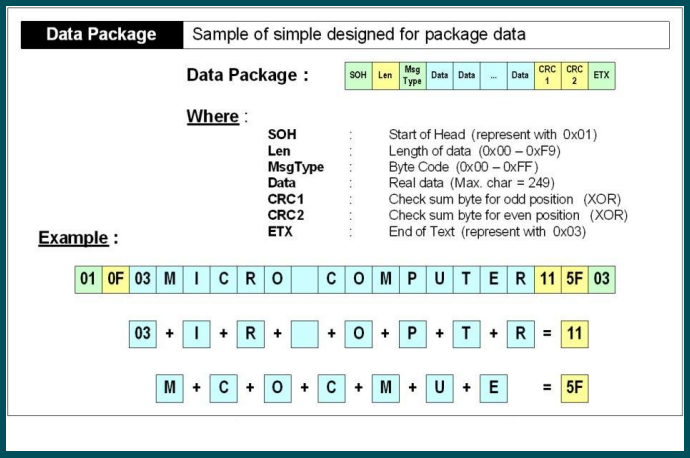
Data packets



Start bits tell the computer when a new data packet is about to begin.
Data bits make up the 'message' or 'body' of the data packet.
The odd parity bit and stop bit make up the 'footer' or 'tail'. These tell
The computer when the data packet is ending.

These data packets are what an older PS/2 mouse sends to a computer.

Data packets



(From: https://www.avrfreaks.net/sites/default/files/Package%20data%20design.jpg)

Where is the header?
Message?
Trailer?

What do you think this data packet does?

Control systems

- In control systems, one is concerned with sensors and actuators to control output functions or actions.
  - An 'actuator' is a component of a machine that is responsible for movement.
  - A 'sensor' is a hardware device designed to respond to changes in the physical environment or machine/device state.
- Open control systems do not react to their environment. A card reader for a hotel door has a sensor that reads the required data off the card, which then activates the actuator to unlock the door if the data matches required conditions.
- Closed control systems react to their environment. Sensors feed back the current machine or environmental state which then adjust or control the actuators as required.

AURORA COLLEGE