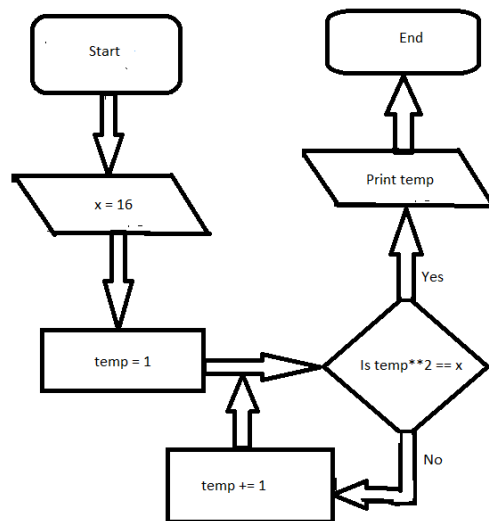


1. ALGORITHMS AND DESIGN

1.1 Fundamental algorithms

1.1.1 concept of a program and flowcharting



Oval = Start / End
 Parallelogram = Input / Output
 Rectangle = Process
 Triangle = Decision

1.1.2 search algorithms such as linear/sequential search, binary search, and hash table search, and comparison of efficiencies

| Search algorithm | Time complexity |
|--|--|
| Linear | Best: O(1) Worst: O(n) |
| Binary FUNCTION bin_search(arr, high, low, x) RETURNS INTEGER mid <- int((high + low) / 2) IF high < low THEN RETURN -1 ELSE IF array[mid] > target THEN RETURN bin_search(arr, high, mid + 1, x) ELSE IF array[mid] < target THEN RETURN bin_search(arr, mid - 1, low, x) ELSE RETURN array[mid] ENDIF ENDFUNCTION | XXXXXXXX XXXX XXXX XX XX XX XX X X X X X X X X Let x be number of steps taken $n = 2^{**}x$ $X = \log(n)$ Best: O(1) Worst: O(log(n)) |
| Hash table <i>Calculate address of target data using the same hash function. If not found, adjust the address depending on method of resolving collision.</i> | Best: O(1) Worse: O(n) |

1.1.3 sort algorithms such as bubble sort, insertion sort, quick sort, and comparisons of their efficiencies

| Sorting algorithm | Time complexity |
|---|--|
| Bubble <pre> PROCEDURE bubble(array) noSwap <- False i <- 0 REPEAT noSwap <- True FOR j = 1 to length(array) - i IF array[j] < array[j - 1] THEN temp <- array[j] array[j] <- array[j - 1] array[j - 1] <- temp noSwap <- False ENDIF ENDFOR i <- i + 1 UNTIL noSwap ENDPROCEDURE </pre> | Best: $O(n)$ Worst: $O(n^2)$ |
| Selection <pre> PROCEDURE selection(array) FOR i = 0 to length(array) - 1 iMin <- i FOR j = i + 1 to length(array) IF array[j] < array[iMin] THEN iMin <- j ENDIF IF iMin <> i THEN temp <- array[i] array[i] <- array[iMin] array[iMin] <- temp ENDIF ENDFOR ENDPROCEDURE </pre> | Best: $O(n)$ Worst: $O(n^2)$ |
| Insertion <pre> PROCEDURE insertion(array) FOR i = 1 to length(array) j <- i - 1 temp <- array[i] REPEAT array[j + 1] <- array[j] j <- j - 1 UNTIL j < 0 OR temp < array[j] array[j + 1] <- temp ENDFOR ENDPROCEDURE </pre> | Best: $O(n)$ Worst: $O(n^2)$ |
| Quick sort <pre> FUNCTION quick(array) RETURNS LIST IF length(array) > 0 pivot <- array[0] left <- [] right <- [] FOR i = 1 to length(array) IF array[i] < pivot THEN left.append(array[i]) ELSE right.append(array[i]) ENDIF ENDFOR RETURN quick(left) + pivot + quick(right) ENDIF </pre> | Best: $O(n \log(n))$ Worst: $O(n^2)$ // Everything either bigger or smaller than pivot, so instead of splitting into 2 lists each time, left/right will be empty while right/left will have just 1 element less each time |

- 1.1.4 modulo operation and weighted-modulus method of computation, and random number generation
- Weighted-modulus: Use weights and modulo of each digit to generate a checksum
 - True random: Random numbers obtained from uncontrollable physical phenomena, like movement of the mouse
 - Pseudo-random: Numbers generated through math formulas or precalculated tables that appear random
- 1.1.5 binary tree traversals: pre-order, in-order and post-order, and hence binary tree sort and binary tree search.
- Pre-order: Output, Left, Right
 - In-order: Left, Output, Right
 - Post-order: Left, Right, Output

1.2 Abstraction

- 1.2.1 data types such as integer, real, char, string, Boolean
- 1.2.2 ASCII codes for character representation and representation of positive integers in binary, octal and hexadecimal forms
- 1.2.3 the Unicode encoding system (that is, the Unicode Standard)
- Unicode allows for transmitting of every language, characters and other symbols, but is inefficient as it requires multiple bytes
 - ASCII (American standard code for information interchange) is a 7-bit (8th bit for parity check) character encoding scheme that is limited to 128 characters
- 1.2.4 data structures such as array, stack, queue, list and binary tree, and their associated algorithms
- Array: puts all data linearly in a storage space
 - Stack: First in last out
 - Queue: First in first out
 - List (linked-list): Each node has a pointer to the next node. Nodes are added or deleted by changing pointer values
- 1.2.5 concept of recursion
- Recursion is a technique that divides the problem into smaller instances of the same problem to solve it
 - 3 characteristics of recursion:
 - Every recursive call reduces the original problem until it reaches the base case
 - Solutions for base cases can be directly provided or calculated
 - Backtrack to arrive at solution of original problem
 - Advantages and disadvantages:
 - Iteration is usually faster than recursion as recursive calls must be stored in a stack first before returning back to the caller functions
 - Iterative algorithms are less resource intensive

- Recursive algorithms are more intuitive to code as they mimic the human thought process. Recursive code is also more easily understood by people other than the original programmer, aiding in code readability, future development and maintenance

1.2.6 limitations of the computation, problem situation and/or computer.

1.3 Modularity

1.3.1 types of program errors: semantic, syntax, logic and arithmetic, and why they occur

- Semantic: Wrong interpretation of program statement (type error)
- Syntax: Code violates grammar rules of programming language
- Logic: Allows program to run successfully but produce unintended results
- Arithmetic: Illegal mathematical operations
 - Overflow: Results of arithmetic operation too large to fit into finite number of bits, leading to wrong or negative answer stored
 - Underflow: Results of arithmetic operation too small to fit into finite number of bits, leading to zero stored

1.3.2 appropriate test cases (normal, abnormal, erroneous and boundary data) for testing algorithms and debugging techniques

- Test cases consist of:
 - (i) normal data: *Normally entered into system, which should accept, process and output results. Ensure that actual output is same as expected output*
 - (ii) boundary data: *Absolute limits of normal data, which should accept, process and output results. Ensure that actual output is same as expected output*
 - (iii) abnormal data: *Invalid and should not be accepted by system. Ensure invalid data does not break system*
- Debugging techniques:
 - Print statements
 - Use print statements to print out values of variables at various execution stages of the code to locate where the error occurs
 - Set breakpoints
 - Set breakpoints and trace the function call stack to check if recursive calls are called with the correct parameter values. This helps detect wrong function calls or wrong parameter values
 - Unit testing (check if actual output matches expected output)





1.3.3 data validation techniques: range and type checks, and the difference between data validation and data verification

- Data validation: Ensure input data values are correct, clean and useful
 - Range check: Check that input values are within a specified range (E.g. 1-9, a-z, A-Z, <space>)
 - Type check: Check if input is of the correct data type
 - Presence check: Check for empty fields

- Format check: Check if the position of every character in the input matches the correct format
 - Length check: Check if the input contains a specific number of characters
 - Check digit: Use the last character of a code to check if the other characters are correct
- Data verification: Check if copy of data is consistent with the original data to ensure data transfer is successful

1.3.4 techniques to formulate and represent a computer solution (e.g. data specification, top-down design, modular design, data flow diagrams, decision table/tree, pseudocode and programming language code) and step-wise refinement where applicable

DFD

| Symbol | Purpose |
|--|---|
|  | Entity (Customer, teacher, user) |
|  | Process (Start with a verb, like 'Check booking', 'Make payment') |
|  | Database (File store, Key information) |
|  | Information flow (Login credentials, Key ID) |

- Modular design: Break down program functions into different modules which each solves a problem by itself

DECISION TABLE

| Conditions | C1 | C2 | C3 | C4 |
|------------------|----|----|----|----|
| Black | Y | N | Y | N |
| Contains liquid | Y | Y | N | N |
| Decisions | | | | |
| Pen | X | X | | |
| Pencil | | | X | |
| Others | | | | X |

REDUCED TO...

| Conditions | C1 | C3 | C4 |
|------------------|----|----|----|
| Black | - | Y | N |
| Contains liquid | Y | N | N |
| Decisions | | | |
| Pen | X | | |
| Pencil | | X | |
| Others | | | X |

UNIFIED MODELLING LANGUAGE CLASS DIAGRAM

| | |
|--|-----------------------|
| Staff | Class name |
| -name -ID -age | Attributes |
| +Staff (name: String, ID: String, age: Integer) +get_name (): String +get_age(): Integer -set_id (ID: String) -set_age(age: Integer) | Constructor & methods |

*Inheritance arrows must have empty arrowheads

*Polymorphism works across parent class and sub class, and not across sub classes. This means that a method that exhibits polymorphism must be present in both parent and child class

*Include inherited attributes, but not inherited methods, except for those that show polymorphism

- 1.3.5 that clarity of programming solution may be enhanced through comments, indentation, white space and meaningful names.

1.4 Programming

1.4.1 input, output, sequence, selection and iteration constructs in programming

1.4.2 serial and sequential text files

- Serial file:
 - Records are stored and ordered in chronological order (unsorted, only by time)
 - E.g. log file
 - Usually linear search
- Sequential file:
 - Records are stored and ordered by key order
 - Since file is sorted according to the key, binary search can be used
- Random access file
 - Records are stored and ordered by applying a hash function to the record keys which gives the address of the record
 - To resolve hash collision, linear or quadratic probing can be used
 - With linear probing, clustering is likely to occur, slowing down the search and insertion time

- Quadratic probing involves adding quadratic values of successive integers to current hash value
 - Files can utilise variable or fixed length records
 - Fixed length allows for quick access as program knows exact location of where each record starts, but this results in larger file sizes as padding is used to standardise record length
 - Variable record length requires less storage space but is slow as program does not know start and end of record (must find delimiters)
- 1.4.3 classes and objects
- Class: Blueprint that defines the attributes and methods of an object
 - Object: Entity created by the class
- 1.4.4 encapsulation and how classes support information hiding and implementation independence
- Encapsulation:
 - Refers to the bundling of data with the methods that operate on them in a single unit, the class.
 - This helps in hiding private variables within the class, preventing public program code from modifying private variables, protecting an object's internal state from corruption by its clients
 - Public methods such as getters and setters are required to access or modify private variables
 - Minimises interdependencies between modules by defining a strict interface. This allows the program code to be changed without affecting the interface if the new implementation supports the same interface. i.e implementation of an object can be changed without affecting the application using it, allowing for bug fixes and maintenance
- 1.4.5 inheritance and how it promotes software reuse
- Inheritance:
 - The ability to create sub classes that adopt the attributes and methods of the super class
 - This promotes code reusability as subclass can adopt data and methods from its superclass, extend it by adding new functionalities or override existing data and methods
- 1.4.6 polymorphism and how it enables code generalisation
- Polymorphism:
 - The ability of different objects to respond to the same method in different ways invoked. i.e Methods with the same name can work differently in different classes

2 INTERFACE AND INTERACTIONS

2.1 Interacting with computers

2.1.1 types of user interfaces (e.g. command-line, menu, form-based, graphical)

- Command-line
 - Input: Command string
 - Output: Printed text on monitor
 - Useful for batch processing jobs like batch renaming, copying, deletion
- Graphical user interface
 - Input: Using keyboard and mouse, interact with graphical elements
 - Output: Graphical outputs
- Form-based
 - Mixture of drop-down lists, radio buttons, check-boxes, text boxes, date field, for user to complete an online form to enter data into the system
 - Used to facilitate user input and perform validation
- Menu
 - Series of menus and submenus
 - Input: User is provided with a set of predefined actions which he can select, and will be provided with another menu to decide next course of action

2.1.2 specifications of appropriate interface and user interaction

- Touch design (iPad, Smart phones)
 - Interface:
 - Large buttons
 - Icons to represent functions to reduce clutter
 - Large front, less text
 - Interaction:
 - Touch and voice commands (Voice reduces need for typing)
- Form GUI design
 - Interface:
 - Text boxes with restricted length
 - Dropdown boxes – select 1 option out of many fixed options
 - Checkboxes – select multiple options
 - Radio buttons – select single option out of few fixed options
 - Labels – remind users of appropriate input format
 - Built-in validation
 - Check for duplicate NRICs
 - Valid numeric input for phone numbers
 - Valid format for email addresses

2.1.3 design considerations for user interfaces

- User familiarity
 - Design of the interface should be based on the experience of the users that will make most use of the system
- Minimal surprise
 - User should not be surprised by behaviour of system
- Consistency
 - Interface should be consistent with similar processes

- Diversity
 - Interface should be suitable for different groups of users:
 - Multiple language selections for people that speak different languages
 - Audio feedback for visually impaired users
- User guidance
 - Provide meaningful feedback when errors occur and context sensitive help facilities
- Error recovery
 - Allow user to recover from errors (Undo button)
- 2.1.4 interaction techniques such as mouse click, key press, use of voice, gesture, and eye movement
 - Mouse clicks
 - Allows user to interact with graphical elements of a GUI
 - Key press
 - Provide input through buttons pre-mapped with commands
 - Voice
 - Gesture
 - Eye movement
- 2.1.5 interaction styles such as command line, menu, graphical user interface and virtual reality
 - Command line
 - [A] Power and flexible as users can achieve different functionalities by specifying parameters. Multiple files can also be processed at once.
 - [D] Steep learning curve, poor retention of command, high error rates, poor error management, not suitable for non-experts
 - Form fill-in
 - [A] Simple data entry, easy to learn, users are guided via predefined rules, facilitates validation
 - [D] User choices may not match form options, sets the scene for rigid procedures
 - Menu selection
 - [A] User can afford to explore, avoids user error, minimal typing, easy to use and learn
 - [D] If too many nested menus, it becomes complex. Slow for experienced users
 - Direct manipulation
 - [A] Intuitive and fast to use, recognition memory
 - [D] Only suitable when there are visual metaphors for actions, hard to implement
 - Natural language
 - [A] Useful when hands are preoccupied, suitable for casual users
 - [D] Lack of support for less widely used languages, unreliable understanding systems
- 2.1.6 social, ethical and economic effects of the use of computers at work, in life and play

2.2 Interfacing computers

- 2.2.1 local area network (LAN), wide area network (WAN) and examples of current technologies
 - LAN: A network that connects devices in over a small geographical area
 - WAN: A network that connects devices over a wide geographical area

2.2.2 the different purposes of the hardware for a network such as servers, clients, switches, routers and bridges

- Servers:
 - A host machine which provides services and resources to clients that request it
- Clients:
 - A client requests services and resources from servers
- Switches:
 - A device that connects multiple devices to form a network
 - Uses packet switching to forward data to destination
 - Only forwards data to devices that need to receive it instead of broadcasting to all devices on the network
- Routers:
 - A device that connects multiple networks together, forwarding packets between computer networks
 - Router exists outside a LAN
- Bridges:
 - A device that connects multiple LAN segments together to form a single network
OR
A device that segments a network into sub segments
 - Bridge exists within a LAN

2.2.3 why organisations use intranets

- Intranet:
 - A private network that is only accessible to a organisation's staff
 - A wide range of information and services of the organisation's internal information technology system is shared within the intranet, which are not accessible by the public through the internet
 - An intranet enhances productivity by:
 - Facilitates collaboration by providing shared workspaces
 - Quick access to information
 - Keep all employees updated
 - Improved communication between employees

2.2.4 concept of cloud computing and kinds of clouds (i.e. application, infrastructure, and platform)

- Internet-based computing that provides shared computer processing resources and data to customers on demand
- Helps companies avoid up-front infrastructure costs (of servers), allowing them to focus on core business plan
- Applications will also be up and running faster, with improved manageability and less maintenance. Along with the ability to view cloud usage patterns, this allows businesses to adjust resources easily to meet fluctuating business demands
- Software as a service
 - Allow clients to use the provider's applications running on cloud infrastructure
 - Eliminates the need for customers to install and run applications on their own devices. Customers only require internet access to use the cloud application
 - E.g. Google apps like google docs

- Platform as a service
 - Provide a platform for clients to deploy their own applications onto cloud infrastructure
 - Clients have control over the deployed app, allowing clients to create customised apps
 - If there are multiple developers working on the same project, PaaS is useful as it gives enormous collaboration capabilities such as having a version control system
 - E.g. Heroku, Google app engine
- Infrastructure as a service
 - Provide clients with control over infrastructure like the servers, storage and networks to deploy and run arbitrary software on cloud infrastructure
 - Clients have full control over the deployed software and the infrastructure without having to physically own, maintain and manage the hardware
 - They are charged based on consumption, making IaaS highly scalable
 - E.g. Microsoft Azure
- Disadvantage of cloud computing:
 - Limited customizability
 - Security and privacy

2.2.5 rate of data transmission (in baud or bps)

- Symbol: Fixed number of bits (bag of coins)
- Baud: Fixed number of symbols (bucket of bags)

2.2.6 synchronous and asynchronous data transmission

- Synchronous: A byte is sent after a standardised time interval
 - [A] Lower overhead
 - [D] Hardware needed for synchronisation is more expensive
 - E.g. Ethernet cables
- Asynchronous: Special start and stop bit patterns are inserted at the beginning and end of each packet.
 - [A] Cheaper than synchronous data transmission as there is no need for hardware for synchronisation which are expensive
 - [D] Relatively large overhead due to start and stop bit patterns which causes a large proportion of transmitted bits to be for control purposes only, and carry no useful information
 - E.g. Telephone lines use asynchronous data transmission as users can talk whenever they want

2.2.7 simplex, half-duplex and full-duplex mode of data transmission

- Simplex: 1-way communication E.g. Radio broadcasting
- Half-duplex: 2-way communication but only one at a time. E.g. Walkie Talkie's press to talk
- Full-duplex: 2-way communication simultaneous. E.g. Ethernet cables contain 2 twisted pairs within the same jacket, one for receiving and one for transmitting data

2.2.8 packet switching and circuit switching for data transmission

- Packet switching:

- Messages are broken down into packets
- The packets are then routed individually through the network
- At the receiver, packets are assembled to construct the original message
- Circuit switching:
 - A dedicated circuit is established for the duration of the transmission

2.2.9 the use of parity checks and checksums in detecting and correcting errors in data transmission.

- Detect errors during transmission of data
- One-bit parity check
 - A bit is appended to the end of each byte to make the total number of '1's either odd or even, corresponding to the chosen parity check
 - To check for corruption of transmitted byte, count the number of '1's, and if the parity of the byte matches the parity set by the packet, then the byte is valid
- Parity block
 - The bytes are arranged in a 2 dimensional form, and the number of bits in every row and column are counted to see if it matches the parity set by the packet itself
- Checksums
 - A number is generated by hashing data values
 - The checksums generated prior to and after transmission are compared to ensure error-free transmission

2.3 Interacting with data

2.3.1 attributes of a database (e.g. tables, records, fields, and tuples)

- Primary key: An attribute or a combination of attributes for which there is a value in each tuple and that value is unique
- Foreign key: An attribute in a table that refers to the primary key in another table
- Relation: A table in a relational database
- Record: Row of data in a table
- Field: Single piece of data of an attribute in a record

2.3.2 E-R diagrams

- Represents the relationships between entities in a relational database

2.3.3 concepts of data redundancy and data dependency

- Data redundancy: A situation where the same data is stored more than once
- Data dependency: The phenomenon where the correct functioning of an application that uses data in a database relies on the way that this data is organised

2.3.4 the use of a database management system

- DBMS is a software that helps create and manage databases. It is used to create, retrieve, update and manage data. It is the middleware between the database and the applications that need to interact with the database

2.3.5 the need for privacy and integrity of data

- Data privacy: The requirement for data to be available only to authorized users

- When sensitive data is required for a system, a privacy plan should be provided to clearly state what the data will be used for, and to make sure that it will not be used for any other purposes.
- Data integrity: The requirement for data to be accurate and up-to-date

2.3.6 Normalisation

- Unnormalized form (UNF)
 - (project code, project title, project manager, project budget, employee 1 number, employee 1 name, department 1 number, department 1 name, employee 2 number, employee 2 name, department 2 number, department 2 name)
- First normalized form (1NF)
 - Attribute values should be in atomic, and there should be no data redundancy
 - Project (project code, project title, project manager, project budget)
 - Resources (project code*, employee number, department number, employee name, department name)
- Second normalized form (2NF)
 - All non-key attributes must depend on knowing all of the primary key
 - Project (project code, project title, project manager, project budget)
 - Employee (employee number, employee name, department number, department name)
 - Project members (project code*, employee number*)
- Third normalized form (3NF)
 - No dependencies between non-key attributes
 - Project (project code, project title, project manager, project budget)
 - Employee (employee number, employee name, department number*)
 - Department (department number, department name)
 - Project members (project code*, employee number*)
- Unnormalized data will have data anomalies:
 - Insertion anomaly
 - Data cannot be inserted due to absence of other necessary data
 - Deletion anomaly
 - Deletion of data leads to unintended loss of related data
 - Update anomaly
 - When data is updated, and the same changes are not made to every record containing the data, data inconsistency results

3 SYSTEMS ENGINEERING

3.1 System development life cycle

- 3.1.1 the data and processes in a software system/computer-based application (e.g. business systems, information systems, education and entertainment systems)
- 3.1.2 the phases in a system development cycle (specification, design, development, documentation, implementation, testing/modification and maintenance)
 - 1) Specifications
 - Gather information using methods like:
 - (a) Questionnaires

- Gather information from many people
 - [A] Easy and cheap to implement, tabulate and analyse data
 - [D] Only able to probe on superficial issues
 - (b) Interview
 - To gather expert and detailed information
 - [A] Interactive, immediate feedback
 - [D] Biased response
 - Research report
 - Existing system review
 - Information of what is being done and how is it done
 - Set of problem statements to describe problems with existing system
 - New system requirements
 - System objectives
- 2) Design
 - Technical design (UDFS – u dun f say, UI, Data store, system Flow, Security)
 - Define system flow and processes by defining system inputs and outputs
 - Define data stores
 - Usability design like user interface
 - Security procedures
 - Programming design
 - Programming approaches
 - E.g Test driven development (TDD) where requirements are turned into test cases, and the system is improved to pass the test cases
 - System development approaches
 - Top-down design: Overview is formulated, then sub-systems are refined until entire system is reduced to its base elements
 - Bottom-up design: Base elements are specified in great detail first, and then linked to form the entire system
 - Modular design: Functions of system are designed separately as modules which can function on their own
- 3) Programming
 - The computing solution is transformed into executable computer programs according to the programming approach specified in the design phase
- 4) Documentation
 - User documentation
 - Contains instructions on how to operate the system, with minimal technical jargon. Prepared by business analyst
 - (i) Purpose of system
 - (ii) Minimum hardware and software requirements
 - (iii) Detailed instructions on how to operate the system, including example inputs and outputs
 - (iv) Troubleshooting guide with explanations of errors and how to manage the errors
 - (v) FAQs
 - (vi) Contacts for further assistance
 - Technical documentation

- Intended for personnel that maintain the system. Prepared by system analyst
 - (i) System specifications
 - (ii) Minimum hardware and software requirements
 - (iii) System flowcharts (DFD)
 - (iv) User interface designs
 - (v) Test plans
- 5) Testing
 - Test plan
 - Test cases selected should test normal, boundary and abnormal data inputs

| Test No. | Test Data | Purpose | Expected Results | Actual Results |
|----------|-------------------------------------|-------------------------|------------------|----------------|
| 1 | Enter correct value X = 1 | Test 'input x' function | Accepted | |
| 2 | Enter incorrect value X = -99999 | Test 'input x' function | Rejected | |

- Top-down / Bottom up testing

| | Description | Advantages | Disadvantages |
|-----------|--|---|---|
| Top-down | Test most important to least important component | Prototype would have main functionalities ready | Lower level components not tested as much |
| Bottom-up | Test least important to most important component | Easier to find bugs Helps to determine level of software development | Long time for prototype to be developed |

- Black box testing (Or system testing)
 - Used to check that output of a program, given certain inputs, would conform to functional specifications of the system
 - Knowledge of the internal structure of the system is unknown to tester
 - A good black box test should test the full range of the program by testing the boundary conditions of the program using boundary values as test values
- White box testing
 - Uses specific knowledge of the program code to test the internal structure of the system
 - Knowledge of the internal structure of the system is known to tester
 - A good white box test should test all possible paths in a program
- Alpha testing
 - Simulation of actual operations tested by potential users at the developer's site
- Beta testing

- System is released to a limited audience outside the developer's team, where testing is done by real users with real data. This helps developers identify potential bugs under realistic conditions
 - User acceptance test
 - End-users are guided through a series of typical task scenarios to test the performance, effectiveness and usability of the software. Done by Quality Assurance manager.
 - Integration testing
 - Individual software modules are integrated and tested as a group to expose defects in the interface and interaction between modules
- 6) Installation

| | Description | Advantages | Disadvantages |
|-----------------------|---|---|--------------------------------|
| Direct changeover | Old system is stopped completely, new system is started. New data is inputted into new system | Takes minimal effort, time and money | No back up |
| Parallel running | Old system is kept running in parallel. New data is inputted into both old and new systems | Old system acts as back up Output from both old and new systems can be compared to ensure that new system is running correctly | Takes extra time and resources |
| Phased implementation | New system is implemented in phases, gradually replacing parts of the old system until new system takes over entirely | Allow users to gradually get used to the new system Training of workers can also be done gradually | No back up |
| Pilot running | New system is piloted in a part of the organisation. If successful, it will then be implemented for the entire organisation | If new system fails, only a small part of the organisation will be affected | No back up |

- 7) Maintenance

| | |
|------------|--|
| Corrective | Maintenance of bugs undetected during testing phase |
| Perfective | Refinement and optimisation E.g. Efficiency for large inputs |

| | |
|------------|--|
| Predictive | Prepare for likely future changes |
| Adaptive | Maintenance to meet changes in requirements and the environment E.g. User need |

- 3.1.3 the need for a test plan using strategies such as bottom-up testing, top-down testing, white box testing, black box testing, and alpha and beta testing.

3.2 Project management techniques

3.2.1 purpose of project management

- Ensure that project meets all criteria for acceptability, budget and deadlines

3.2.2 the purpose of a project proposal

- A request for financial assistance to implement a project
- Convince people to take part in the project
- Serves as a guide in implementing the project
- Components: (PART-B, Purpose, Activity, Resource, Team, Budget)
 - Project aims – explain core problem and why project is important
 - Resource plan with resource allocation consideration
 - Activity plan
 - PERT and Gantt chart for project progress management
 - Budget
 - Qualifications of team members

3.2.3 project management processes and tools (e.g. critical path analysis and Gantt chart)

- 5 main phases in project management:
 - 1. Initiate
 - Identify objective of project and stakeholders
 - Develop a Project Charter to authorize the start of the project
 - 2. Plan
 - Establish an activity and resource plan
 - Responsibilities are assigned
 - Risk management plan is developed, which identifies different possibilities of arranging tasks, as well as point out tasks with the most risk that require careful monitoring
 - 3. Execute
 - Write program codes
 - Perform testing
 - 4. Monitor and control
 - Conduct review meetings to monitor project progress
 - Monitor project progress using against the critical path
 - Prepare progress reports and present on project progress to the team
 - Take correct measures like re-schedule activities or acquire extra resources as needed
 - 5. Closure and evaluation
 - Project files are archived
 - All deliverables are signed off
 - Assess customer satisfaction
 - Document lessons learned

- Evaluate performance
- Project management tools
 - Gantt chart
 - [A] Shows the time period over which tasks should be completed
 - [A] Can be easily visually interpreted to help keep track of the progress
 - [D] Size of bar does not indicate amount of resources needed. Misinforms the size of the task.
 - [D] Chart may need to be constantly updated due to delays or unforeseen circumstances
 - [D] Does not show task dependencies
 - PERT chart
 - [A] Shows tasks dependencies, allowing critical path to be identified
 - [A] Can be easily visually interpreted to help keep track of the progress
 - [D] Resource intensive as a detailed study of project tasks is needed, making it expensive to develop a PERT chart
 - Dummy activities are imaginary activities with zero completion time that represent dependencies between different activities that would otherwise be hard to show with simple arrow linkages

3.2.4 the importance of team work and the roles of team members working on a computer project

- Importance of team work
 - Maximise individual strengths, team members complement each other
 - Manage workload
- Roles
 - Project manager
 - Develops project plan with the team
 - Responsible for project delays
 - Sign off deliverables
 - Coordinate resources
 - Monitor and present on project status
 - Ensure project is delivered with all criteria for acceptability, deadlines and budget met
 - System analyst
 - Design all technical aspects of the system
 - Modifies and evaluates program code as needed
 - Prepare technical documentation
 - Coordinate system testing
 - Monitor technical issues that arise during implementation
 - Business analyst
 - Prepare business aspect of user documentation
 - Prepare research report which contains user requirements
 - Understand business requirement
 - Set the direction for test plans
 - Quality assurance manager
 - Conducts user acceptance test with client
 - Developer

- Code according to programming design (includes programming approach and system development approach)

3.3 Network applications

3.3.1 methods and tools for creating a network application (e.g. client-server scripting/ programming, hand-held devices, technology standards and application software)

- Client-side scripting:
 - Executed on client's side, by user's web browser
 - Scripting languages include HTML, JavaScript, CSS
- Server-side scripting:
 - Technique used in website design which involves embedding scripts in HTML source code which results in a user's request being handled by a script running server-side
 - Uses include processing of user input, displaying web pages
 - Scripting languages include PHP, ASP
- Client-server network:
 - A network design where tasks and workloads are partitioned between clients and servers. Clients refer to software or hardware that requests resources and services from a server, while a server refers to a host machine that provides services and resources.

3.3.2 methods for ensuring security of a network application (e.g. access rights and password access)

- Authentication: Users verify their identity using a set of credentials
 - 2 Factor Authentication: An additional layer of security where user provides 2 different authentication factors to verify their identity
- Authorisation: Control the operations and resources that an authenticated client can access
- Encryption: Protect sensitive data in transit over the network. Does not protect data integrity, only its confidentiality. E.g. Secure Sockets Layer (SSL) is a security protocol that establishes encrypted links between the webserver and the browser
- Configuration: Security settings and user access controls must be configured properly, limiting the operations and resources that an authenticated user can access. This ensures that different users have the appropriate level of access to the data. For example, doctors should only have the patient records of those under their care, and not of all patients in the hospital.
- Update: Keep services updated to prevent exploitation of security vulnerabilities found in outdated services
- ~~Digital signatures~~
- System can be set to disconnect the user after a period of inactivity to prevent unauthorized users from viewing sensitive information when the employee is not around
- Surveillance cameras in server room
- Sensitive data can be stored only in hardcopy

3.3.3 network security for network applications (e.g. firewalls)

- Network threats:
 - Viruses, worms, trojan horses can open up vulnerabilities in the computer, allowing the perpetrator to gain root access to the computer
 - Denial of service attacks, identity theft
 - SQL injection on data-driven applications allows the attacker to disclose data in the database and / or tamper with existing data

- Cross-site scripting is the injection of malicious JavaScript code in a website that can change the way the website looks and behaves
- Firewall:
 - A network security system that monitors and control both incoming and outgoing network traffic based on predetermined security rules
 - Restricts access to certain ports of the computer, preventing hackers from gaining information about services listening on the ports, which present vulnerabilities for hackers to exploit. By only allowing common ports like port 22 for SSH, 80 for HTTP, hackers only have limited range of vulnerabilities to work with, reducing chances of penetrating the computer
 - Also drops packets from untrusted external sources
- Intrusion detection system:
 - Monitors network and system activities for malicious activities, such as vulnerability scanning

3.3.4 copyright, ethics and social issues involving network applications.

- Patent:
 - The rights granted to the owner of an invention which can be a product or process that gives a technical solution to a problem.
 - It must have some form of practical application in some form of industry
- Copyright:
 - Grants exclusive rights to control the use and distribution of expression of ideas that have an element of originality
- Copyleft:
 - Author surrenders some, but not all rights under copyright law
 - Attribution
 - Share-alike
 - Non-commercial
 - No derivative works