1(a)
i) Copy contents of PC to MAR

(ii) Fetch instruction at the address indicated in MAR and place in MDR

(iii) Copy instruction from MDR to CIR

(iv) Increment PC

(v) Decode Instruction

(vi) Repeat the above steps in the right sequence above.

1(b):

Program counter (PC) – it holds the address of the next instruction to be fetched.

Memory Address Registrar (MAR) – It holds the address of the memory location of the instruction to be executed.

Instruction Registers (IR) or called Current Instruction Register (CIR) – It stores instruction fetched from memory to be executed.

Memory Data Registrar (MDR) – It stores instruction fetched from memory before moving to CIR.

1 (c i) Computer viruses are software programs that are designed to spread viruses from one computer to another and to interfere with computer operation or even create damages.

(c ii) Kind of damages that could be caused are:

- Virus might corrupt or delete data on your computer, use your e-mail program to spread itself to other computers, or even erase everything on your hard disk.
- Virus may cause the system to shut down.
- Confidential data may be stolen if the system is effected by virus attack.

(c iii) Two precautions (one mark for each point, max 2 marks)

- Viruses are most easily spread by attachments in e-mail messages or instant messaging messages. Thus avoid opening e-mail attachments from unknown sources.
- Viruses can be disguised as attachments of funny images, greeting cards, or audio and video files on some web-site, thus avoiding downloading files from these unknown sources. Be aware of some free ware that may consist of infected files.
- Make use of anti-virus software to scan the system regularly for viruses.
- Install firewall which blocks any unauthorized web site and illegal connections.

2(a)

<LETTER>::= A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z

<FRONTCODE>::= <DIGIT><LETTER>|<LETTER><DIGIT>|<DIGIT><DIGIT>

<DIGITS> ::= <DIGIT>|<DIGIT><DIGITS>

<FLIGHT>::= <FRONTCODE><DIGITS>

2(b)

<NONZERO> ::= 1|2|3|4|5|6|7|8|9
<FLIGHT> ::= <FRONTCODE><NONZERO>|<FRONTCODE><NONZERO><DIGITS>

3(a)
(i) Encapsulation: The combination of data and functions into a single entity
(ii) Object: an instance of a class with data and associated functions
(iii) Inheritance: the ability for sub-classes or child classes to be derived from a more general class (called the super class or parent class) where the attributes and methods of the parent class are "inherited" without the need to recode or re-specify
(iv) Polymorphism: ability of functions of the same name to behave differently in different contexts

3(b)

Class name →
Data attributes →
- private
Methods →
+ public

Class Student
- ID - Name
+ Student() + getID() + setID(string) + getName() + setName(string) + display()

3(c)
- Closer mapping to real world interactions, hence more intuitive in design when creating solutions to real-world problems
- Reusability: instantiation and reuse of objects
- ability to create new classes from existing classes through inheritance

4(a)

- Methods of searching

  Binary search: the number of items to be searched keeps halving until the item is found, or when it can be concluded that the item is not present.

  Sequential search: each and every item is looked at in sequence, until the item is found or when all the items have been looked at.

- The list needs to be sorted before implementing binary search, whereas sequential search can be implemented without sorting the list.

(b) Assuming that array is sorted in ascending order

```
BinarySearch (array, target, first, last)
If (first > last )
        Report "item not found"
    Else
      Middle = (first + last) / 2
    If (array [middle] = target)
```

```
                Report “item found at position” middle
                Else
                    If (array[middle] > target)
                        binarySearch (array, target, first, middle – 1)
                    Else
                    binarySearch (array, target, middle + 1, last)
                    Endif
                Endif
            Endif
```

(c)

```
    Bubble Sort
        set i = 5, Flag=0
repeat
            for j=1 to i-1 do
                if X[j] > X[j+1] then
                  swap X[j] and X[j+1]
                  Flag = 1
                endif
            endfor

            if Flag = 0 then Exit

                else set i=i-1

            flag =0  // next pass

    until i=2
```

Trace table

pass 1	6112	3218	2685	9164	7413	Flag =0
	3218	6112	2685	9164	7413	Flag = 1
	3218	2685	6112	9164	7413	Flag = 1
	3218	2685	6112	9164	7413	Flag = 1
	3218	2685	6112	7413	9164	Flag = 1
Pass 2						Flag = 0
	2685	3218	6112	7413	9164	Flag = 1
	2685	3218	6112	7413	9164	Flag = 1
	2685	3218	6112	7413	9164	Flag = 1
Pass 3	2685	3218	6112	7413	9164	Flag = 0
	2685	3218	6112	7413	9164	Flag = 0

(c) The steps are:

The quicksort algorithm is a recursive sorting algorithm based on the divide and conquers approach.

How it works?

Pick an element, called a pivot, from the list. (Usually pick the first element in the list as pivot)

Using a Partition routine to reorder the list so that all elements which are less than the pivot come before the pivot and all elements greater than the pivot come after it (equal values can go either way), Thus forming a left sublist and a right sublist.

After this partitioning, the pivot is in its final position.

The outcomes of first Quick Sort call will be:

2685 3218 **6112** 7413 9164

(2685, 3218) 6112 (9164, 7413) ---------- Divide

Then Quicksort the left sublist, Recur

Also Quicksort the right sublist

Until all sub-lists of length one are achieved

The whole list (2685 3218 6112 7413 9164) has been sorted----------Conquer

5(a)

The System Development Life Cycle

Detailed definition of the problem

Without a detailed definition of the problem we are trying to solve, people are often unsure of what it is they are trying to achieve. Therefore it is important for us to define the problem.

A feasibility study

A preliminary investigation is essential to determine whether a project is technically, operationally and economically feasible.

Collecting information about the proposed system

Assuming that the project is feasible, much information needs to be collected -- the system's requirements can then be determined in more detail, and more detailed estimates of the likely costs are undertaken.

Analysis

An analysis of the problems using techniques such as top-down approach, and making use of other structured methods now follows.

System Design

This stage involves detailed design of the whole system, input, process and output subsystems. At the end of System Design, an important document system specification is produced.

Coding of all the modules of the project, followed by extensive program debugging, program testing and system testing (black board and white board testing, alpha and beta testing)

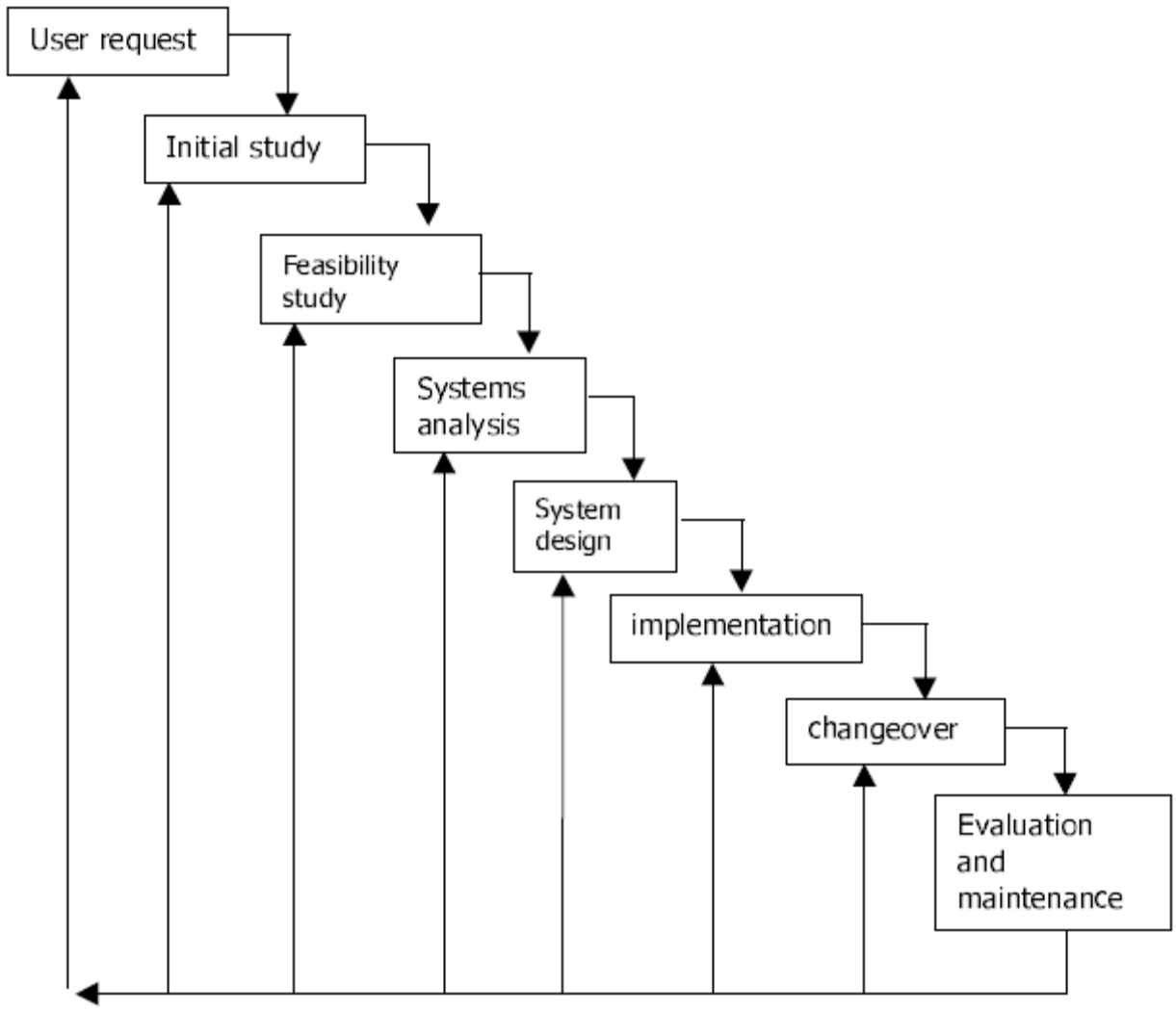
Implementation and evaluation

Implement the developed new system using one of the changeover procedures (direct, phased, parallel and pilot). Staff training takes place during this phase.

Maintenance

Making sure that the system continues to function correctly, and correctly bugs that may come to light after extensive use in the field.

For Revision: A model of a system development life cycle is shown below.



5(b)The System Requirements Specification (SRS) document describes all data, functional and behavioral requirements of the software under production or development.

Content of the System Specification

1. Introduction
   1.1 System summary
   1.2 Objectives of the system
   1.3 Hardware environment
   1.4 Software environment

2. **System description**
   2.1 Narrative description
   2.2 System flowcharts

3. Input
   3.1 Summary
   3.2 Input document descriptions and layout
   3.3 Input media descriptions and layout
   3.4 Input controls

4. Files
   4.1 Summary
   4.2 File descriptions
   4.3 Record layouts
   4.4 File controls

5. Output
   5.1 Summary
   5.2 Output descriptions and layout
   5.3 Report mock-ups
   5.4 Handling and distribution procedures
   5.5 Output controls

6. Special logic requirements
   (Tables, formulae, algorithms)

7. Glossary of mnemonics and special terms used.

6(a)
Three advantages are:

- Improved data consistency
- Reduced data redundancy
- Improve data security

6(b)

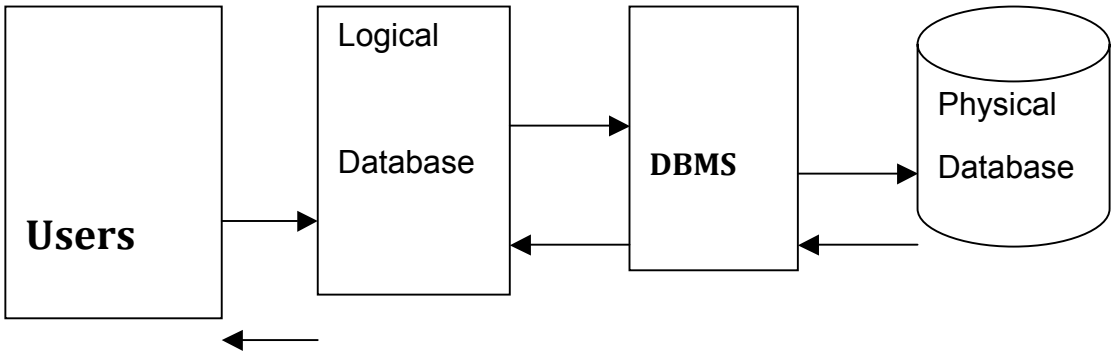
(i) query language-- A facility of a DBMS which allow users to ask for the information to be extracted. Query language is a simplified programming language, restricted to querying a database. Commonly used functions are selecting, searching, interrogating etc.

(ii) report generator -- To generate reports for presenting data that is abstracted from the database. It allows users to define report definition, report format, report layout, display order etc.

(iii) generator for input and output screens -- It allows database users to generate input form on screen for input new data to the database and to generate output display on the screen for on-line query.

(iv) Macros – is a feature of DBMS which allows users to automate process that performs a specific task. It is an imbedded set of commands or instructions that can be activated to process a specific task without user's intervention during the process.

6(c)



A DBMS provides the ability for many different users to share data and process resources. As there can be many different users accessing the same database and submit different requests, DBMS has to provide three views of the database data: an external view (or user view), logical view (or conceptual view) and physical (or internal) view. The user's view of a database program represents data in a format that is meaningful to a user and to the software programs that process those data.

Application data model is a form of logical view which refers to the way the user views the data on screen and the user interacts with the data in this interface. DBMS provides facilities for users to modify the data. The user does not have the knowledge where the data is actually stored. The DBMS links the virtual data shown on screen and the real location of data in the physical storage.

The physical organisation refers to the way the data are physically stored and processed. It is manipulated and maintained by the DBMS which present the physical data model in its logical view for users.

Application data model is a conceptual or logical view that provide users view that is external view, whereas physical organisation is a physical or internal view of the data.

Application data model shows the logical view, i.e. what data is needed by the applications, whereas physical organization shows how data is stored and accessed on physical devices such as hard disk, server etc. In the case of application data model, data is perceived as having the following information: name, type and length of field. In the case of physical organization, data is perceived as being physically located at certain disk, sector, block, etc.

7(a)

```
head-->|Lim|-->|Ong|-->|Tan|-->NULL
free --> |   | --> |   | --> NULL
```

7(b)

```
Insert(Data)
newnode->data = Data
newnode->next = NULL
If List is empty or newnode->data < head->data
  head = newnode
Else
     temp = head
     while (newnode->data > temp->next->data)
        temp = temp->next
     end-while
     newnode->next = temp->next
     temp->next = newnode
End-if-else
```

7(c)

Before attempting to insert the Data, use the Search() method to see if the Data already exist in List, if not, proceed to insert the Data, otherwise disallow

```
If Search(Data) = 0
  Insert(Data)
End-If
```

7(d)

	Data	Next
1	Tan	0
Free = 2		4
Head = 3	Ong	1
4		5
5		6
6		7
7		0

7(e)
Let A store the Search results of the letter of the alphabet required by user, e.g. Search('A')
Let B store the Search results of the letter of alphabet after the letter entered by user, e.g. Search('B').
The total no. of entries not including the single upper-case letters would be: (B – A – 1)

8(a)

```
InsertBST(Tree, Newnode)
If  (Tree is NULL)
    Insert Newnode in position
Else
    If (Tree->Data = Newnode->Data)
        Print error: cannot have duplicates, exit
    Else if (Newnode->Data < Tree->Data)
            InsertBST(Tree->Left, Newnode)
          Else
              InsertBST(Tree->Right, Newnode)
```

8(b)
Add additional field to newnode record structure to store the frequency of occurrence |Data/**Freq**/Next| instead of |Data/Next|
In the InsertBST algorithm, when there is a match, increment the Frequency by one before exiting function, i.e.

```
If (Tree->Data = Newnode->Data)
        Tree->Freq = Tree->Freq + 1
        Print error: cannot have duplicates, exit
```

8(c)

1. Traverse BST by pre-order traversal
2. Store Word and frequency in an array
3. Loop through array
4. Cumulatively add up the frequency
5. Sort array in descending order of frequency
6. Output the total no. of words and the first three elements of the array as the top three words with highest number of occurrences.

8(d)

```
Inorder(Tree)
If Tree is NOT Null
   Inorder(Tree->Left)
   Print Tree->Data
   Inorder(Tree->Right)
End-If
```