**LAB # 04**

**Objectives**

* ***Introduction to DFD.***
* *To familiar with DFD models.*
* ***Data Flow Diagrams: discovering and design the cases.***

**DESCRIPTION:**

Data analysis attempts to answer four specific questions:

What Processes make up a system?

What Data are used in each process?

What Data are stored?

What Data enter and leave the system?

Data drive business activities and can trigger events (e.g. new sales order data) or be processed to provide information about the activity. Data flow analysis, as the name suggests, follows the flow of data through business processes and determines how organisation objectives are accomplished. In the course of handling transactions and completing tasks, data are input, processed, stored, retrieved, used, changed and output. Data flow analysis studies the use of data in each activity and documents the findings in data flow diagrams, graphically showing the relation between processes and data.

**Physical and Logical DFDs**

There are two types of data flow diagrams, namely *physical data flow diagrams* and *logical data Flow diagrams* and it is important to distinguish clearly between the two:

**Physical Data Flow Diagrams**

An implementation-dependent view of the current system, showing what tasks are carried out and how they are performed. Physical characteristics can include:



Names of people



Form and document names or numbers



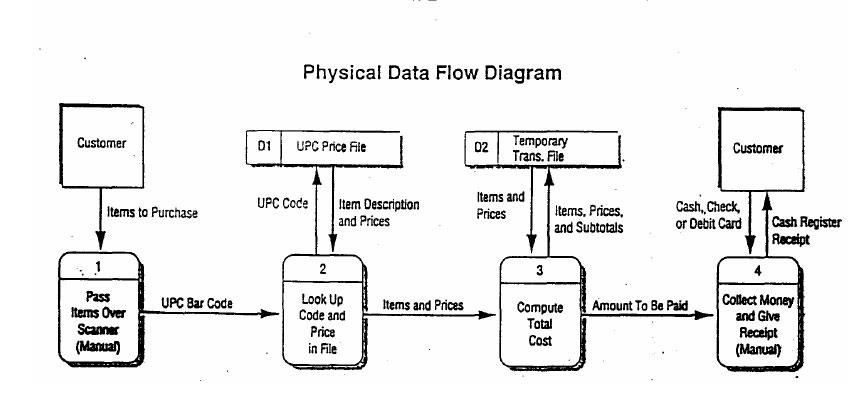
Names of departments



Master and transaction files



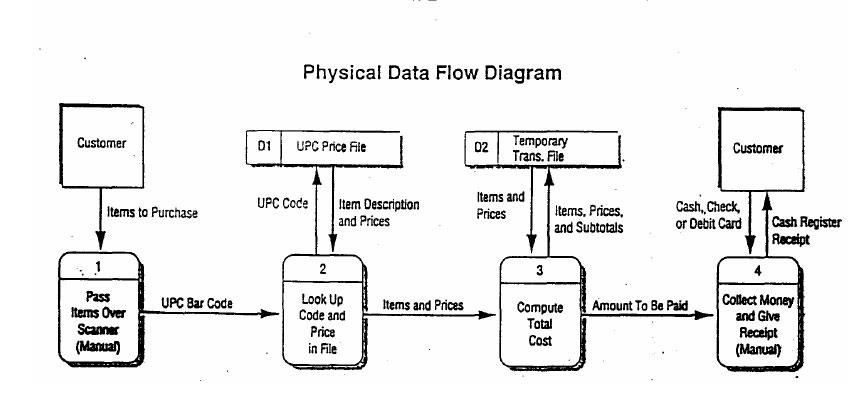
Equipment and devices used



**Logical Data Flow Diagrams**

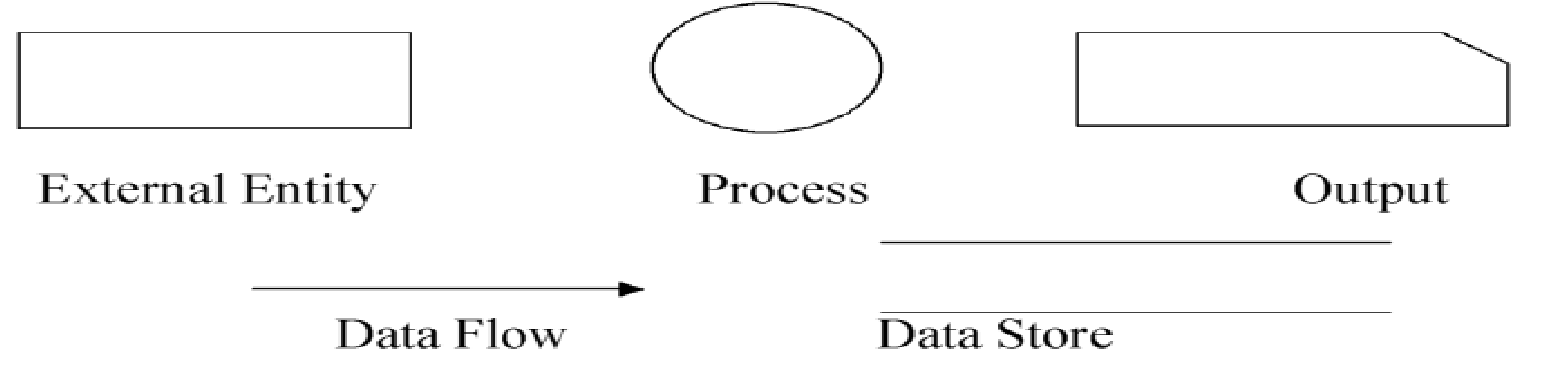
An implementation-*independent* view of the system, focusing on the flow of data between processes without regard for the specific devices, storage locations or people in the system. The physical characteristics listed above for physical data flow diagrams will not be specified.

A logical DFD focuses on the business and how the business operates. It describes the business events that take place and the data required and produced by each event.



**Data Flow Diagram (DFD)**

The DFD (also known as a bubble chart) is a hierarchical graphical model of a system that shows the different processing activities or functions that the system performs and the data interchange among these functions. Each function is considered as a processing station (or process) that consumes some input data and produces some output data. The system is represented in terms of the input data to the system, various processing carried out on these data, and the output data generated by the system. A DFD model uses a very limited number of primitive symbols to represent the functions performed by a system and the data flow among these functions.

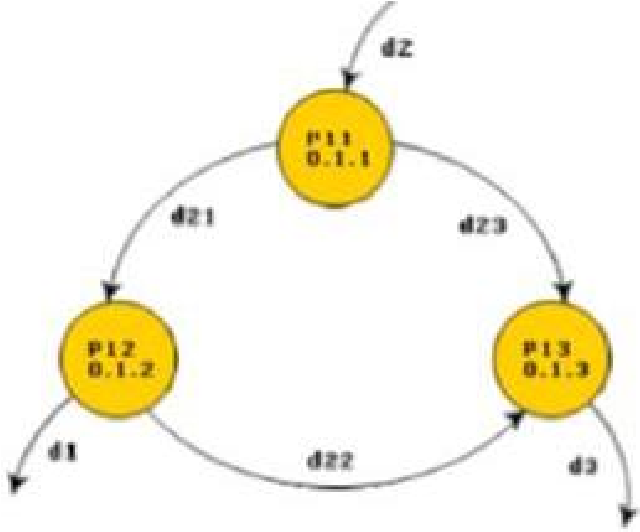
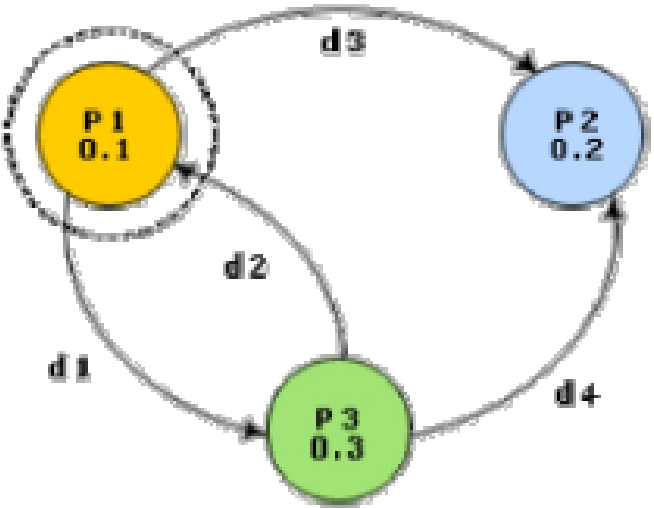


**Importance of DFDs in a good software design**

The main reason why the DFD technique is so popular is probably because of the fact that DFD is a very simple formalism – it is simple to understand and use. Starting with a set of high-level functions that a system performs, a DFD model hierarchically represents various sub functions. In fact, any hierarchical model is simple to understand. Human mind is such that it can easily understand any hierarchical model of a system – because in a hierarchical model, starting with a very simple and abstract model of a system, different details of the system are slowly introduced through different hierarchies. The data flow diagramming technique also follows a very simple set of intuitive concepts and rules. DFD is an elegant modeling technique that turns out to be useful not only to represent the results of structured analysis of a software problem, but also for several other applications such as showing the flow of documents or items in an organization.

**Balancing a DFD**

The data that flow into or out of a bubble must match the data flow at the next level of DFD. This is known as balancing a DFD. The concept of balancing a DFD has been illustrated in fig. bellow. In the level 1 of the DFD, data items d1 and d3 flow out of the bubble 0.1 and the data item d2 flows into the bubble 0.1. In the next level, bubble 0.1 is decomposed. The decomposition is balanced, as d1 and d3 flow out of the level 2 diagram and d2 flows in.



**Decomposition**

Each bubble in the DFD represents a function performed by the system. The bubbles are decomposed into subfunctions at the successive levels of the DFD. Decomposition of a bubble is also known as factoring or exploding a bubble. Each bubble at any level of DFD is usually decomposed to anything between 3 to 7 bubbles. Too few bubbles at any level make that level superfluous. For example, if a bubble is decomposed to just one bubble or two bubbles, then this decomposition becomes redundant. Also, too many bubbles, i.e. more than 7 bubbles at any level of a DFD makes the DFD model hard to understand. Decomposition of a bubble should be carried on until a level is reached at which the function of the bubble can be described using a simple algorithm.

**Numbering of Bubbles**

It is necessary to number the different bubbles occurring in the DFD. These numbers help in uniquely identifying any bubble in the DFD by its bubble number. The bubble at the context level is usually assigned the number 0 to indicate that it is the 0 level DFD. Bubbles at level 1 are numbered, 0.1, 0.2, 0.3, etc, etc. When a bubble numbered x is decomposed, its children bubble are numbered x.1, x.2, x.3, etc. In this numbering scheme, by looking at the number of a bubble we can unambiguously determine its level, its ancestors, and its successors.

**Data dictionary**

A data dictionary lists all data items appearing in the DFD model of a system. The data items listed include all data flows and the contents of all data stores appearing on the DFDs in the DFD model of a system. A data dictionary lists the purpose of all data items and the definition of all composite data items in terms of their component data items. For example, a data dictionary entry may represent that the data **grossPay** consists of the components regularPay and overtimePay.

**grossPay = regularPay + overtimePay**

For the smallest units of data items, the data dictionary lists their name and their type. Composite data items can be defined in terms of primitive data items using the following data definition operators:

**+:** denotes composition of two data items, e.g. **a+b** represents data a and **b.**

**[,,]:** represents selection, i.e. any one of the data items listed in the brackets can occur. For example, **[a,b]** represents either **a** occurs or **b** occurs.

**():** the contents inside the bracket represent optional data which may or may not appear. e.g. **a+(b)** represents either **a** occurs or **a+b** occurs.

**{}:** represents iterative data definition, e.g. **{name}5** represents five **name** data.

**{name}\*** represents zero or more instances of **name** data.

**=:** represents equivalence, e.g. **a=b+c** means that **a** represents **b** and **c.**

**/\* \*/:** Anything appearing within **/\*** and **\*/** is considered as a comment

**Well-structured**

Well-structured designs improves the maintainability of the system. A structured system is one that is developed from the top down and modular, that is, broken down into manageable components. The modules should be designed so that they have minimal effect on other modules in the system. The connections between modules are limited and the interaction of data is minimal. Such design objectives are intended to improe system quality while easing maintenance task. So, Structure design transform the results of structured analysis (i.e., a DFD representation) into a structure chart.

**Purpose of Structure Chart**

A Structure Chart is a design tool that visually displays the relationships between program modeles. It shows which modules with a system interact and also graphically depicts the data that are communicated between various modules.

Structure charts are developed prior to the writing of program code. They are not intended to express procedural logic a task left to flowcharts and pseudo code. Nor do they describe the actual physical interface between processing functions. Instead, they identify the data passes existing between individual modules that interact with one another.

various modules making up the system

module dependency (i.e. which module calls which other modules) parameters passed among different modules



**Basic building blocks of structure chart**

**1. Rectangular box:**

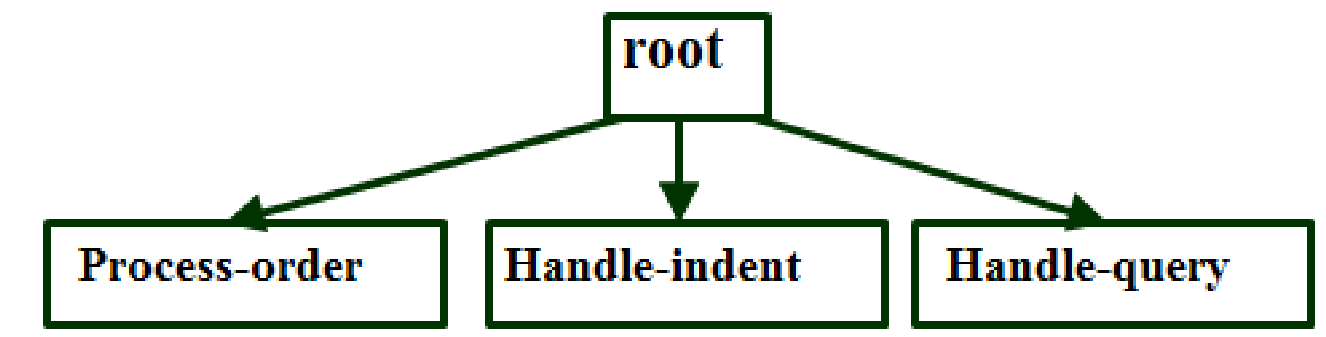
A rectangular box represents a module.

Annotated with the name of the module it represents

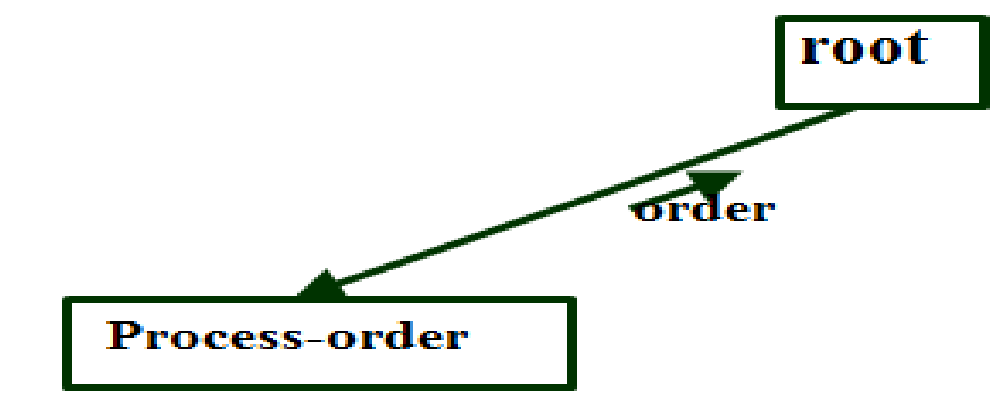


1. **Arrows**

**An arrow between two modules implies:** during execution control is passed from one module to the other in the direction of the arrow.

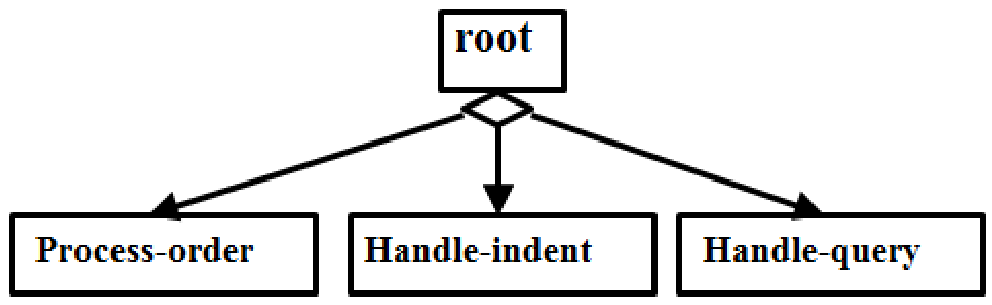


**Data flow arrows represent:** data passing from one module to another in the direction of the arrow.



1. **Diamonds**

**The diamond symbol represents:** one module of several modules connected to the diamond symbol is invoked depending on some condition

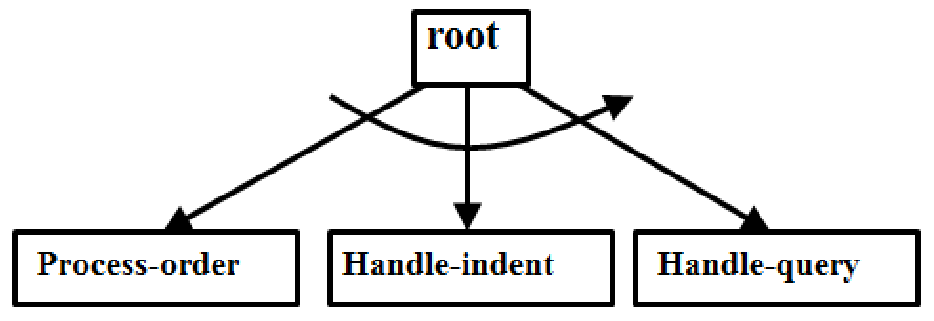


1. **Repetition**

**A loop around control flow arrows denotes:**

that the concerned modules are

invoked repeatedly.



**Guidelines for design**

There is only one module at the top:

 the root module.

There is at most one control relationship between any two modules: if module A invokes module B, module B cannot invoke module A.

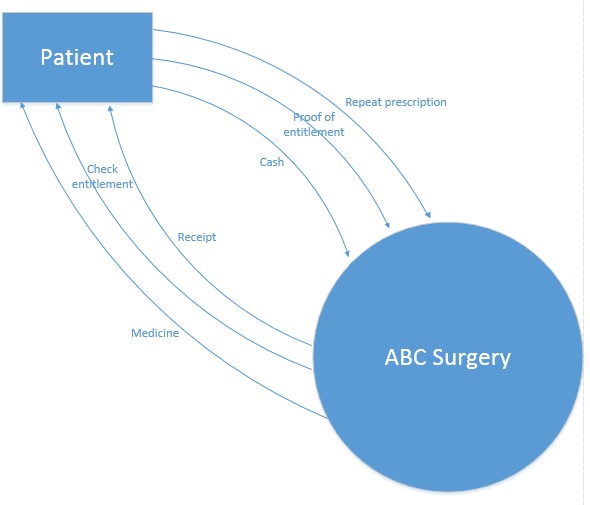
The main reason behind this restriction:

 consider modules in a structure chart to be arranged in layers or levels.

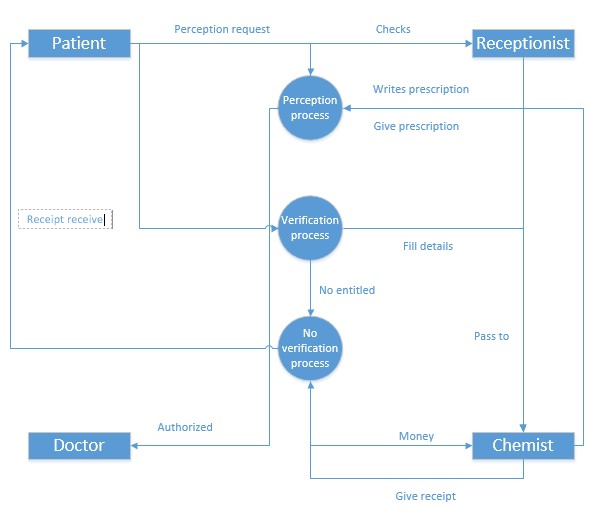
**EXERCISE:**

**Objective:** Creating Data Flow Diagram

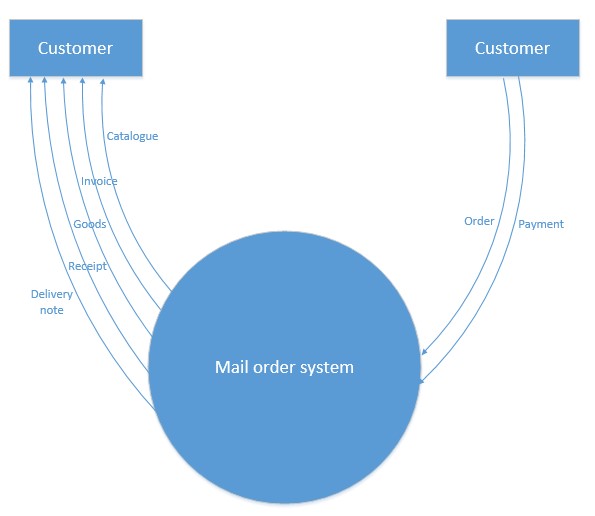
1. Examine the following outline and then draw the diagram you think will model the situation. When a patient arrives at ABC Surgery with a repeat prescription request, the receptionist checks the prescription file and writes out a prescription. This has to be authorized by the doctor before being passed to the resident chemist for dispensing. The chemist then gives the prescription to the patient. If the patient is entitled to free prescriptions, the chemist verifies this and fills in the appropriate details on a form, which is filed in the free prescriptions file. Otherwise the chemist takes the appropriate amount of money from the patient and gives them a receipt.
2. **Level0:**

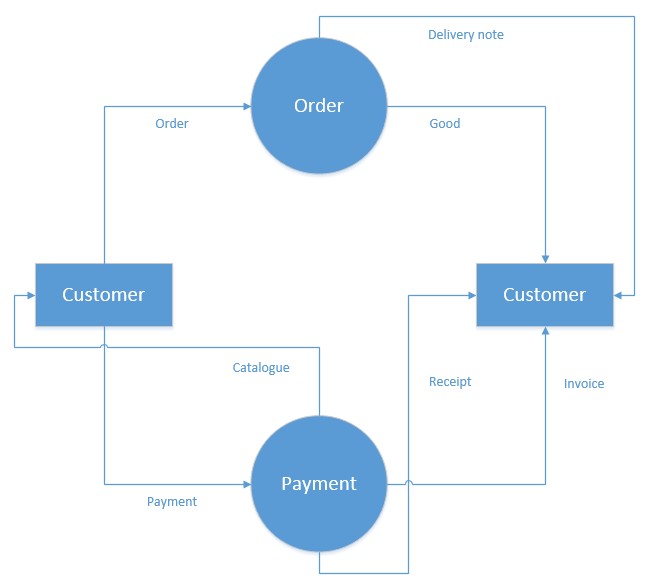


1. **Level1:**



1. Suppose you are given the details of a small mail order catalogue system that allows people to shop from home. When a customer receives the catalogue and wants to buy something, they can telephone, fax or email their order to the company. The company gets the order and sends the goods and an invoice. When the customer receives the goods with a delivery note, they send payment and receive a receipt for their payment. **a. Level0:**

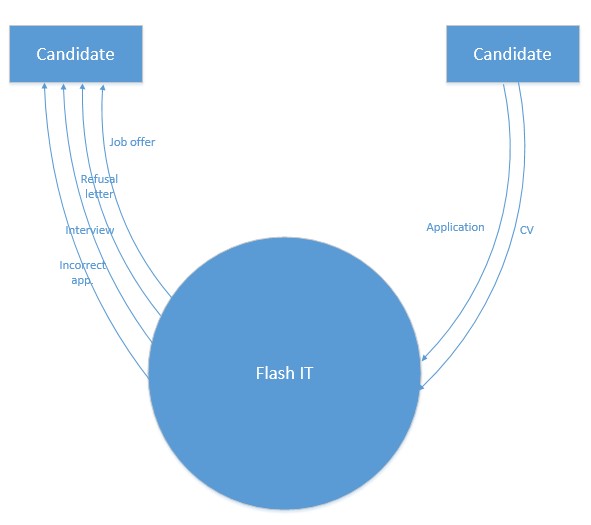
 **b. Level1:**



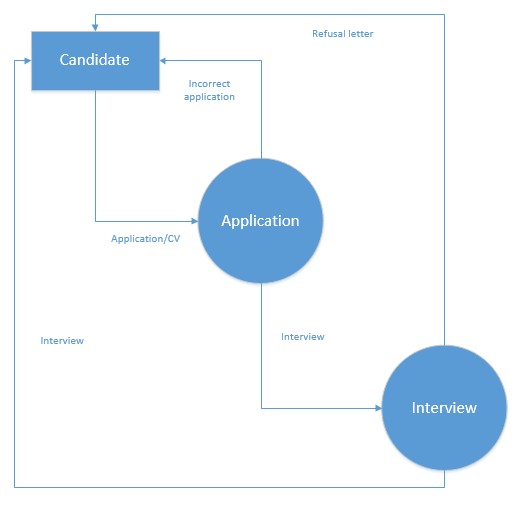
1. When FlashIT are interviewing and selecting new employees for their company, they ask applicants to send their application forms and their CVs to personnel. The personnel department then checks these forms for completeness and, if found to be complete, they are stored in the applications file. Otherwise these forms are returned to applicants for resubmission. Any candidates not considered suitable for the post are sent a refusal letter. Suitable candidates are requested to come in for interview. After interviews have taken place, a decision on the most suitable candidate is taken and they are offered the post.

The interviewees who have been unsuccessful are sent a refusal letter.

1. **Level0:**

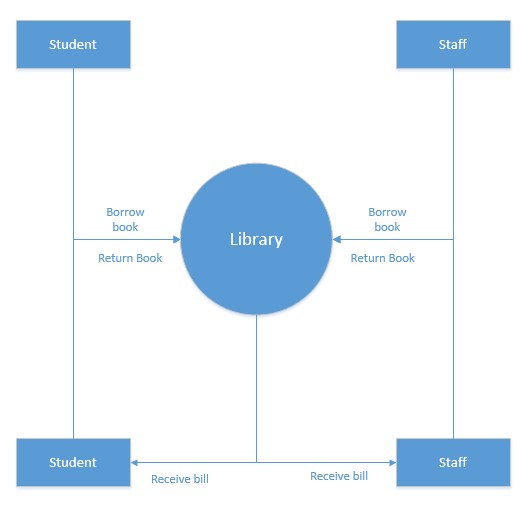


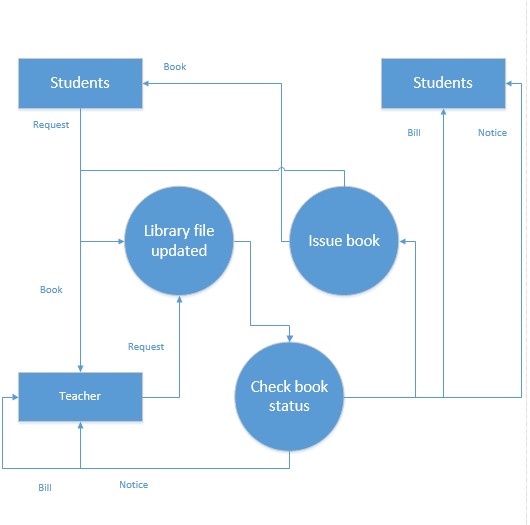
1. **Level1:**



4. XYZ High School has a library that lends books to staff and students. Students are allowed to borrow six books and teachers are allowed to borrow ten. When someone borrows a book the library book file is updated, as is the borrower file. Everyone issued with a book has it for a period of one month, after which time they are sent a reminder. If, after six months, they haven't returned the book, they are sent a bill for the cost of recovery of the book.

**a. Level0:**

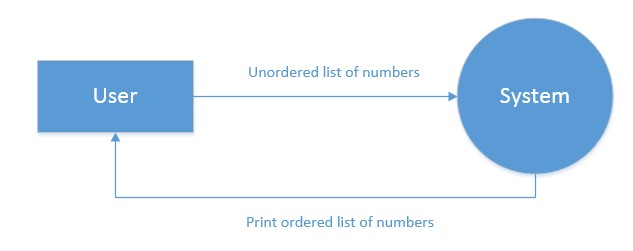
 **b. Level1:**

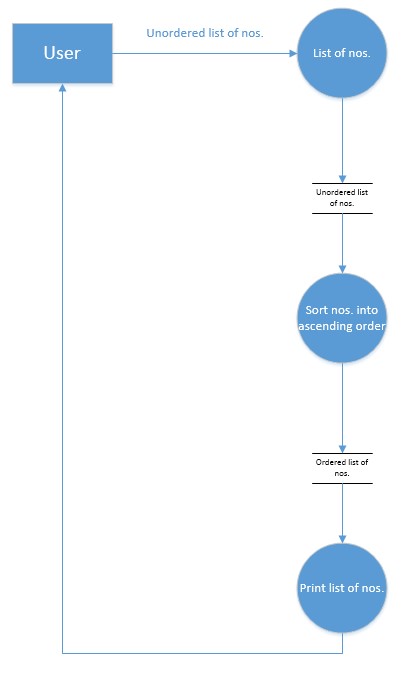


5. A system is required which allows a user to input an unordered list of integer numbers into a computer. The system will store these numbers in the main store of the computer where they are to be sorted by the system into ascending numeric order and re-stored.

Finally the system is to print out the list for the user.

**a. Level0:**

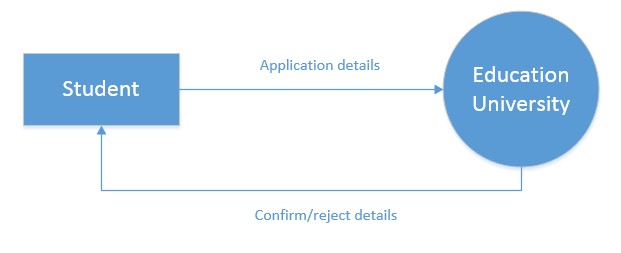
 **b. Level1:**



6. Draw the DFD for Education University. The enrolment process works as follows:

Students send in an application form containing their personal details and their desired course The University checks that the course is available and that the student has necessary academic qualifications. If the course is available the student is enrolled in the course, and the university confirms the enrolment by sending a confirmation letter to the student. If the course is unavailable the student is sent a rejection letter.

**a. Level0:**

 **b. Level1:**

