Name Bilal Ahmad

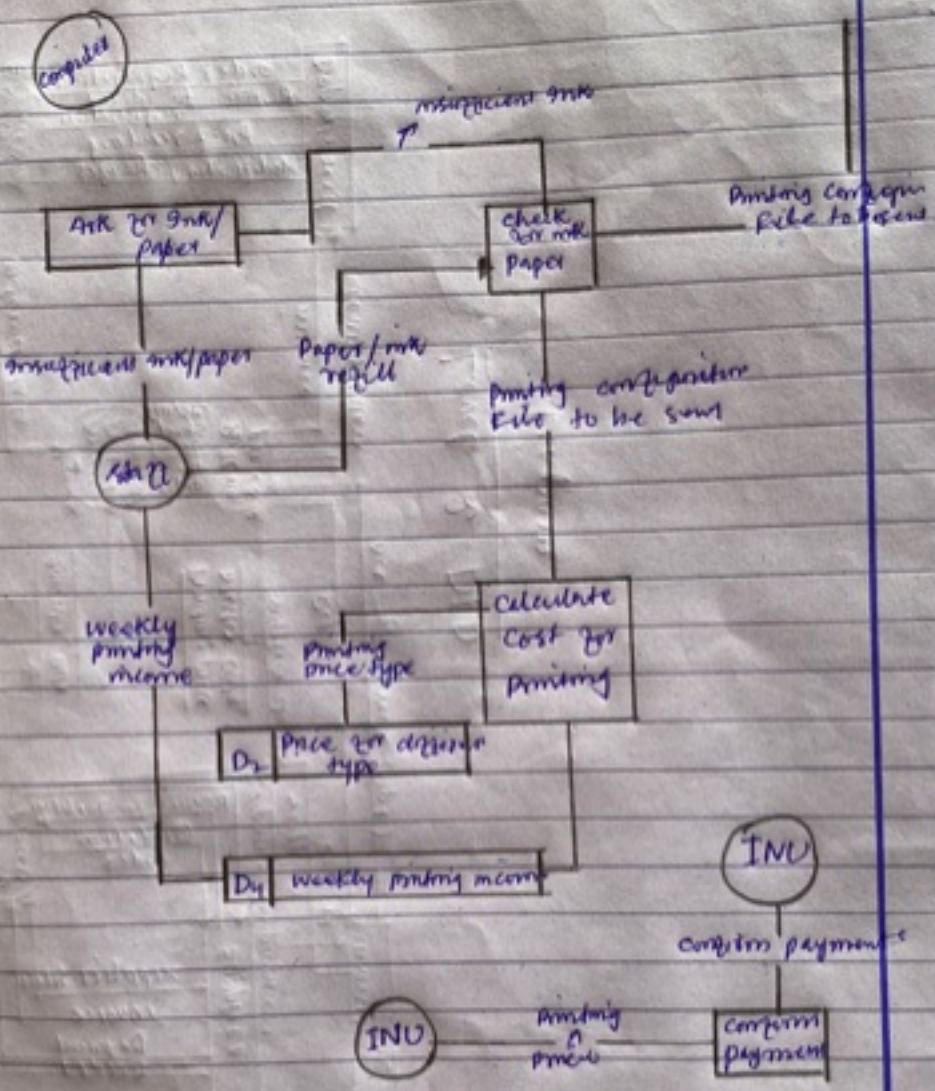
ID = 15089

Subject = Software Engineering

Dept = BSSE

Q1.1:

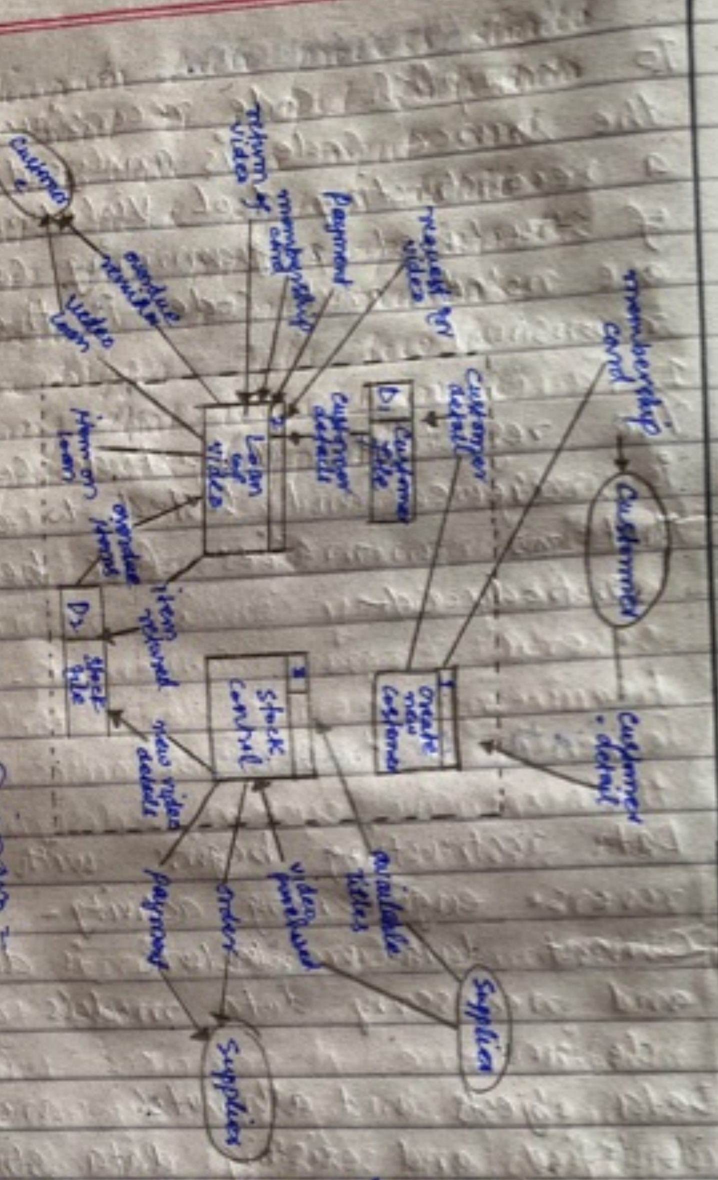
Ans:-



INU printing press Diagram.

Q1.2:-

Ans:-



Level 1 Data flow Diagram:-

A level 1 data flow diagram:-

A level 1 Dataflow diagram notates each of the main sub-processes that together form the complete system. We can think of level 1 DFD as an "exploded view" of the context diagram.

Constructing level 1 DFDs:-

If no context diagram exists, first create one before attempting to construct the level 1 DFD.

The following steps are suggested to aid the construction of DFD.

1) Identify processes each data flow into the system be received by a process. For each data flow into the system examine the documentation about the system and talk to the users to establish a plausible process of system the receives data flow.

2) Draw the data flows between the external entities and process.

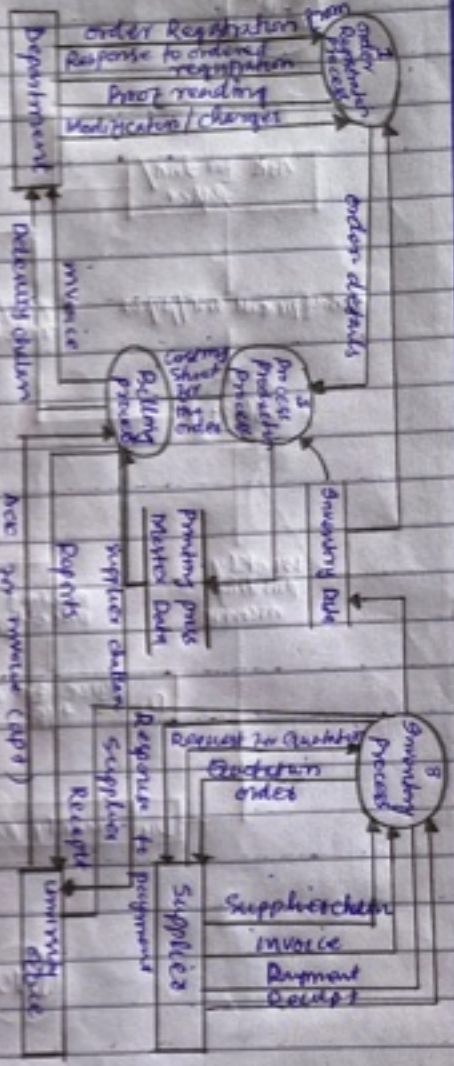
3) Identify data stores by establishing where documents/data need to be held within system. Add the data stores to the diagram. Labelling them with their local name or description.

4) Add data flows flowing between processes and data stores within the system. Each data store must have at least one input data flow; and one output data flow.

5) Check diagram Each process should have an input & output. Each data store should have an input and an output. Check system details so see if any process appears to be happening for no reasons- i.e some trigger data flow is missing that should make the processes happen."

Q1.3 Level 2 DFD

Ans:-



Second level DFD:-

(1) The above DFDs were logical in nature where interaction b/w process was shown considering that a process sending information to another process does so only after processing the information received.

(2) The given diagram is based on the actual situation where information once recieved by a process is stored in data store.

Question 2:

Q 2.1= Explain why testing can only detect the presence of errors, not their absence?

Ans= Testing can detect only the presence of errors, not their absence because the main goal of the testing is to observe the behaviour of the particullar software and to check whether it meet its requirement expectation or not.
It always possible that a test have overlooked could discover further problem with the system.

Q2.2= Define the following terms:

1) Unit testing:-
In computer programming Unit testing is a software testing method by which individual unit of source code, set of one or more computer program modules together with associated control idea data, usage procedures and operating procedures, are tested to determined whether they are fit for use.

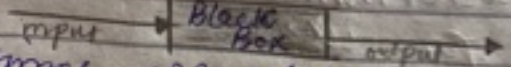
(2) System Testing :-

System testing is a level of testing that validates the complete and fully integrated software product. The purpose of a system test is to evaluate the end to end system specification. Usually the software is interface is only one element of larger computer-based system. Ultimately the software is interfaced with other software/hardware system.

System testing is actually a series of different tests whose sole purpose is to exercise the full computer-based system.

(3) Black Box Testing :-

Black Box testing is a method of software testing that examines the functionality of an application without peering into its internal structures or workings. This method of testing can be applied virtually to every level of software testing unit integration, system and acceptance.

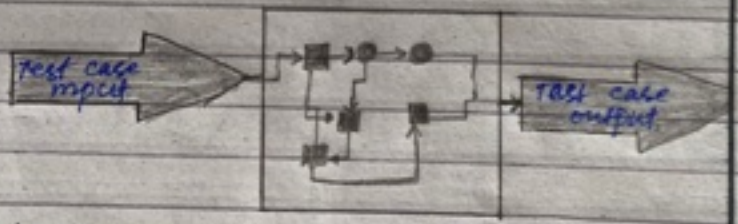


It is sometimes referred to as specification-based testing.

(4) White Box testing :-

White Box testing of a software solution's internal structure, design, and coding. In this type of testing the code is visible to the tester. It focuses primarily on verifying the flow of inputs & outputs through the application, improving design and usability, strengthening security. White Box testing is also known as clear Box testing. Open Box testing, structural testing, Transparent Box testing, code-Based testing & Glass Box testing.

It is usually performed by developer.



White Box testing Diagram.

Question 3:

Q3.1: Briefly describe the three main types of software maintenance. Why it is sometime dizzicult to distinguish b/w them?

Ans:

1) Fault repairs:

Coding errors are usually relatively cheap to correct design errors are more expensive as they may involve rewriting several program components. Requirement errors are the most expensive to repair because of the expensive system redesign which be necessary.

2) Environmental adaptation:

This type of maintenance is required when some aspect of the system's environment. such as the hardware the platezorm operating system. or other support. Software changes the application system must be modified to adapted it to cope with these environmental changes.

(3) Functionality addition:-
This type of maintance is necessary when the system requirement change in response to organizational or business change. The scale of the changes required to the software is often much greater than for the other types of maintance.

Difficulty to distinguish b/w Them:-
In practice there is not a clear-cut distinction b/w these type of maintenance.

Corrective maintenance:-
It is universally used to refer to maintance for fault repair.

Adaptive maintenance:-
Sometimes means adapting to new environment and sometime means adapting the software to new requirements.

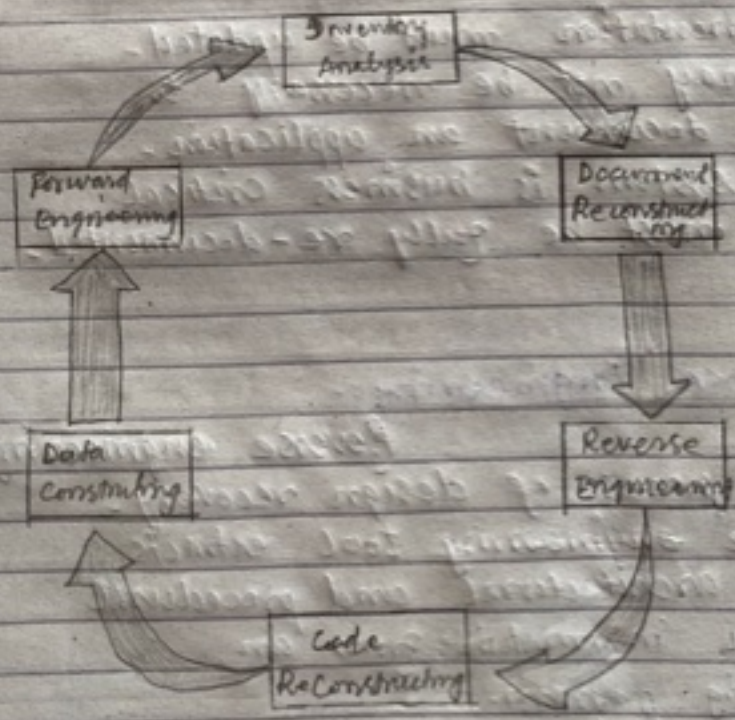
Perfective maintenance:-
Some times perfecting the software by implementing new requirements.

Q3.2:
Ans:

Software Re-Engineering is the examination and alternation of a system to reconstitute it in a new form. The principle of Re-Engineering when applied to the software development process is called re-engineering.

Re-Engineering Costs factors:-

- The quality of the software to be re-engineered.
- The tool support availability for engineering Extent of the data conversion which is required.
- The availability of expert staff for Re-engineering.



quality ... its overall

Software Re-Engineering Activities:-

1) **Inventory Analysis:-**
Every software organisation should have an inventory of all the applications.

- Inventory can be nothing more than a spreadsheet model containing information that provides a detailed description of every active application.

2) **Document reconstructing:-**
Documentation of a system either explains how it operate or how to use it.

- Documentation must be updated.
- It may not be necessary to fully document an application.
- The system is business critical and must be fully re-documented.

3) **Reverse Engineering:-**
Reverse engineering is a process of design recovery. Reverse engineering tool extracts data, architectural and procedural design information from an existing program.

(4) Code Reconstructing :-

- To accomplish code reconstructing the source code is analysed using a reconstructing tool. Violations of structured programming construct are noted and code is then reconstruct.
- The resultant restructed code is reviewed and tested to ensure that no anomalies have been introduced.

(5) Data Restructing :-

- Data restructing begins with the reverse engineering activity.
- Current data architecture is dissected and necessary data models are defined.
- Data objects and attributes are identified and existing data structure are reviewed for quality.

(6) Forward engineering :-

Forward engineering also called as renovation or reclamation not only recovers design information from existing software but uses this information to alter or reconstitute the existing system in an effort to impove its overall quality.