APP CT 3 IMP

1. Logical Programming Paradigm

- 1st class function

- Higher order function

- Pure Function

2. Symbolic Programming Paradigm

3. Network Programming

- TCP & UDP

4. Independent Type Programming

- Quantifiers

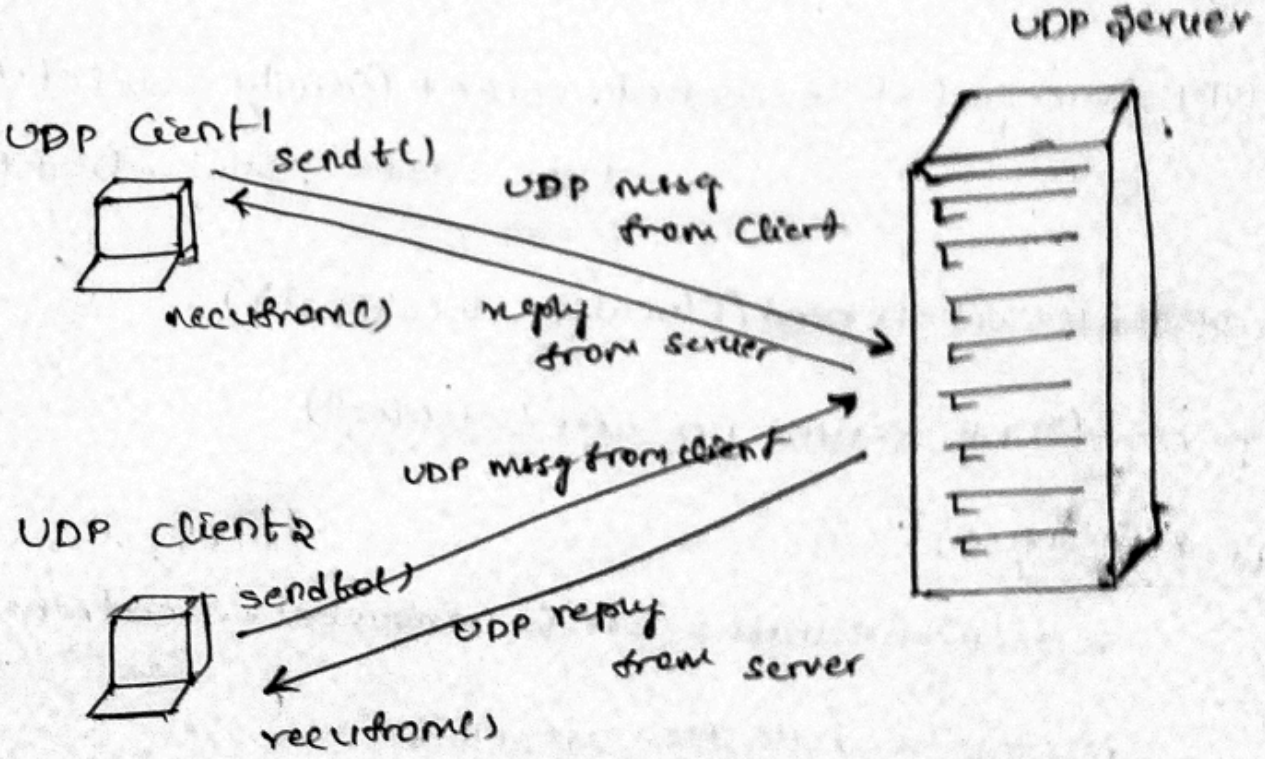
5. GUI Scenario based (PPT / Lab Programs)

6. PPT programs

UDP SERVER:

UDP is the abbreviation of User Datagram Protocol. UDP makes use of internet protocol of TCP/IP suite.

In communications using UDP, a client program sends a message packet to a destination server wherein the destination server also runs on UDP.



Example:-

Program:-

```
import socket
localIP = "127.0.0.1"
localPort = 20001
bufferSize = 1024

msgFromServer = "Hello UDP client"
bytesToSend = str.encode(msgFromServer)

UDPServerSocket = socket.socket(family = socket.AF_INET,
                                type = socket.SOCK_DGRAM)

UDPServerSocket.bind((localIP, localPort))

print ("UDP Server up and listening")

while (true):

    bytesAddressPair = UDPServerSocket.recvfrom(buffer
                                                Size)
    message = bytesAddressPair[0]
    address = bytesAddressPair[1]

    clientMsg = "Message from client: {}".format(message)
    clientIP = "Client IP Address: {}".format(address)
    print (clientMsg)
    print (clientIP)
```

Output:-

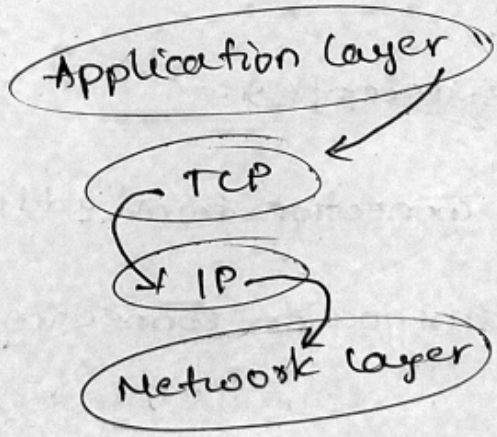
```
UDP Server up and listening
Message from client: b"Hello
                      UDP Server"

Client IP address: ("127.0.0.1", 51980)
```

TCP:- (Transmission Control Protocol)

It is one of the main protocols of Internet protocol suite.

It lies b/w the Application and Networklayers which are used in providing reliable delivery services.

It is a connection oriented protocol for Communications that helps in exchange of messages b/w the different devices over a network.



Working:-

→ TCP breaks the down the data into small bundles & then into the original message on the opposite end.

→ Sending information in little bundles makes it simpler to maintain.

Program:

```
import socket
s= socket.socket()
print ("Socket Successfully created")
port = 12345
s.bind ('', port)
print ("Socket binded to %s %(port))
S.listen(5)
 print ("Socket is listening")
while True:
   c, addr =s.accept()
print ("Got connection from", addr)
c.send ("Thankyou for connecting")
 c.close()
```

Output:-

```
Socket Successfully created
Socket binded to 12345
Socket is listening
Got connection from ('127.0.0.1', 52617)
```

GUI Scenario based :-

GUI stands for graphical user interface.

There are some main interfaces in GUI for developing.

Tkinter Tkinter is the python interface to the Tk GUI toolkit shipped with python

wxPython: This is an open source python interface

Jpython:

It is a python port for Java which gives python concepts.

ALGORITHM:

1) import the tkinter module.
2) Create the GUI application main window.
3) Add one or more of the above-mentioned widgets to the GUI application
4) Enter the mainevent loop to take action against each event triggered by user
5) Stop.

Widgets:-

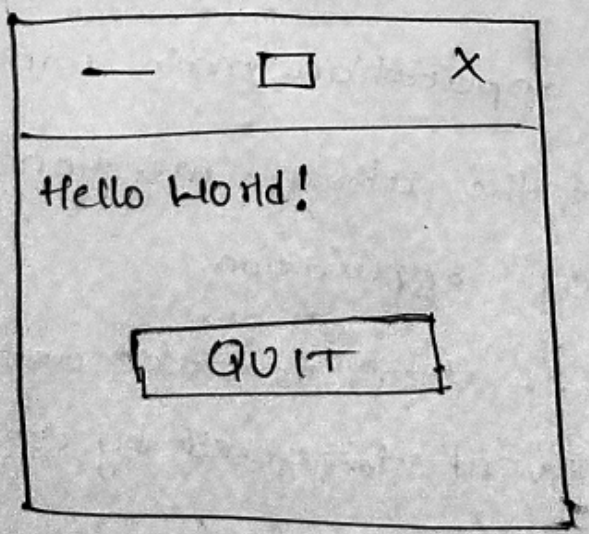
- Label
- Canvas
- Checkbutton
- Listbox
- Button
- Radio button
- Frame
- Menu.
- Message.
- Scale
- Text
- Toplevel.
- Scrollbar

Programs:-

```
import tkinter as tk
root = tk.Tk()
label = tk.Label(root, text = "Hello world")
label.grid()
root.mainloop()
```

```
root2 = tk.Tk()
root2.title(Example)
button = tk.Button(root, text
        = 'Stop', width = 25)
button.pack()
root2.mainloop()
```

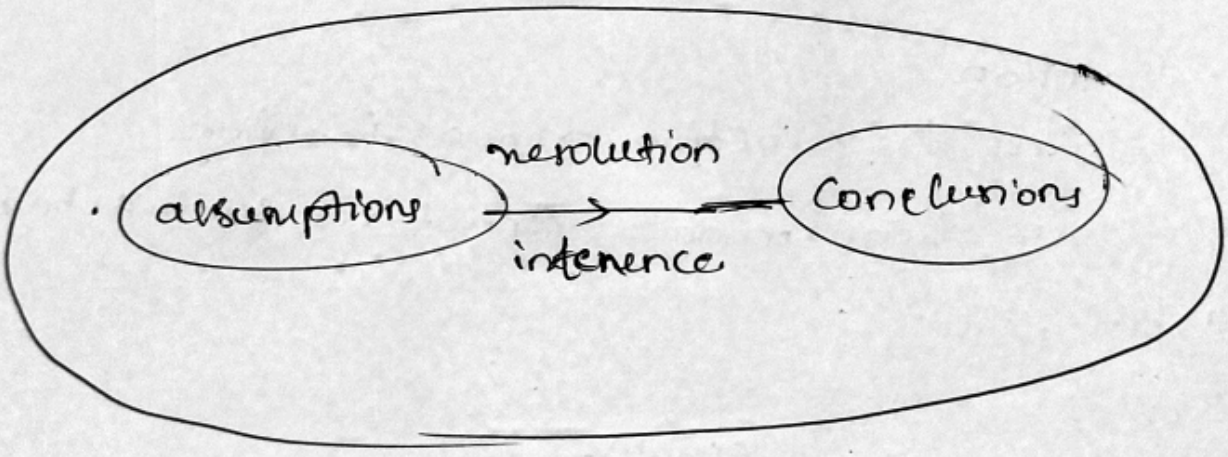
Outputs:-



Logical programming paradigm

Logical programming is a programming paradigm that sees computation as automatic reasoning (theorem) over a database of knowledge made of facts and rules.

A logic is a language. It has syntax and semantics



Syntax:-

~~The rules have to about how to formed~~

The rules about how to form formulas, usually the easy part of logic

Semantics :-

the meaning carried by the formula, usually the easy part of a logic.

First class function :-

→ A function is an instance of the object type.

→ Stores the function in a variable

→ Passes the function as a parameter to another function

→ Returns the function from a function.

→ Stores them in data structures such as hash tables, lists.

Ex:-

```
def shout(text):
    return text.upper()

print(shout('Hello'))

yell = shout
print (yell ("Hello"))
```

output :-

```
HELLO
HELLO
```

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

Higher order function :-

Functions that can accept other functions as arguments are also called high-order functions.

Ex:-

```
def shout(text):
    return text.upper()

def whisper(text):
    return text.lower()

def greet(func)
    greeting = func("Hi, I am created by
                     function passed as
                     argument.")

    print(greeting)
```

Output:-

greet(shout)

greet(whisper)

Pure function:-

A function is called pure function if it always returns the same result for same argument values.

Examples:-

pure functions are strlen(), pow(), sqrt() etc.

Ex:-
impure functions are printf(), rand(), time() etc.

Ex:-

```
time3 = add_time (time1, time2)
print_time(time1)
print_time (time2)
print_time (time3)
```

dependent type programming :-

In Computer Science and logic, a dependent is a type whose definition depends on a value. It is an overlapping feature of type theory and type systems.

→ two common examples of dependent types are dependent functions and dependent pairs.

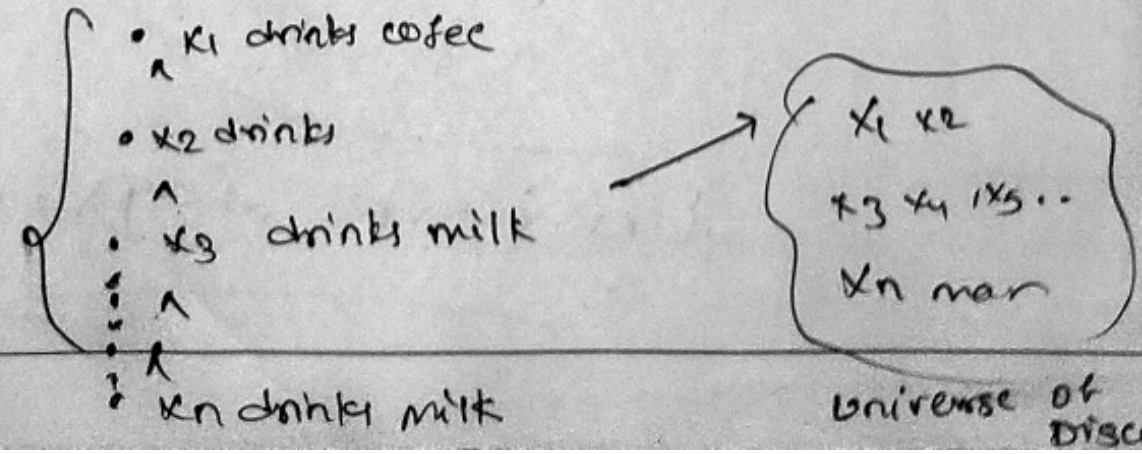
→ The return type of a dependent function may depend on the value of one of its arguments.

Quantifiers :-

All man drink coffee

let a variable x which refers to a cat

So all x can be represented in UOD as below.



$\forall x$ man(x) $\rightarrow$ drink (x, coffee).

Examples :-

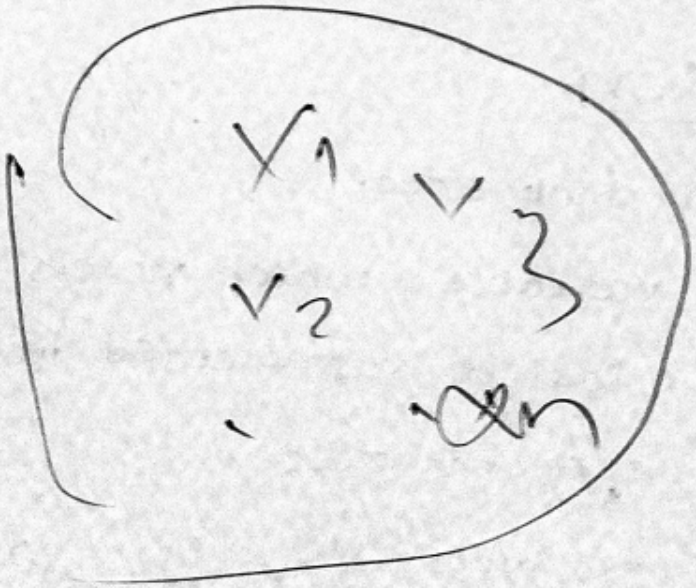
1. All birds fly.

On this question the predicate is "fly (bird)."

And since there are all birds who fly so. it will be represented as follows.

$\forall x$ bird (x) $\rightarrow$ fly(x)

DIAGRAM :-



Universe of Discourse