**DC Practicals**

Name : Aditya Shivram Mahajan

Branch : Computer Science

Reg No. : 2017BCS001

Roll No. : A10

Year : Final Year

## horizontal line

Index

Practical 1 : Design and development of a Distributed application using socket. - [link](#kix.5e1f0q3ikowu)

Practical 2 : Design and development of a threaded Distributed application using socket. - [link](#kix.ilkmf9k77ddb)

Practical 3 : Installation and study of Mobile agent using Aglet platform. - [link](#kix.d6tzr3tk5lsd)

Practical 4 : Design and development of a Distributed application using Agelt platform. - [link](#kix.1r6y9w8rk37g)

Practical 5 : Design and development of a Distributed application using RPC. - [link](#kix.n8858gtar8p6)

Practical 6 : Design and development of a Distributed application using RMI. - [link](#kix.hr5zvjxvr6o)

Practical 7 : Study and Installation of Apache Web Server. - [link](#kix.abey96cspu9c)

Practical 8 : Installation and study of Hadoop Platform. - [link](#kix.dgtvw8xzxhv3)

Practical 9 : Installation and study of Hadoop Cluster. - [link](#kix.nqj48ckmmw6x)

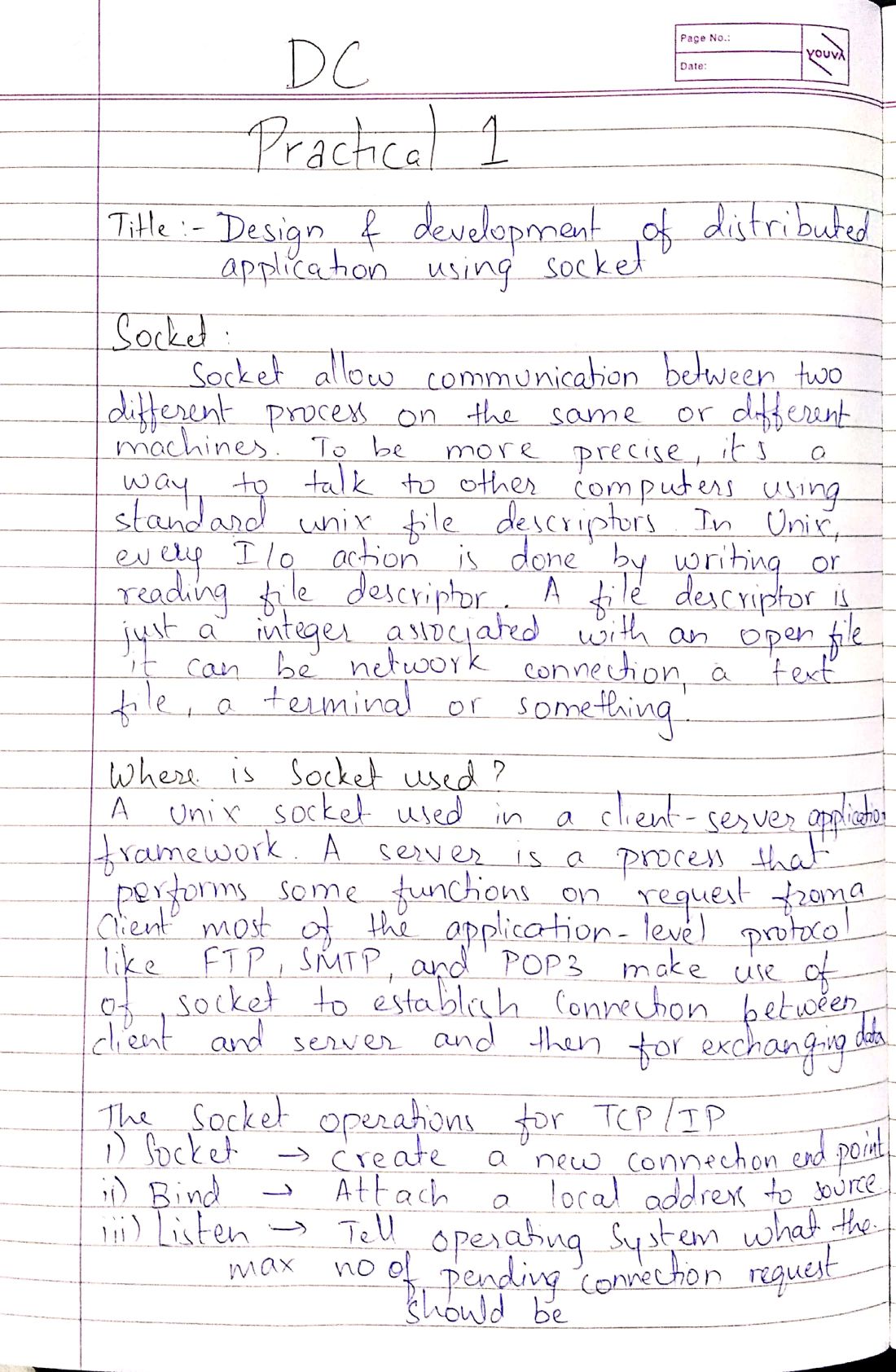
Practical 10 : Study of MapReduce Application to process weather report dataset. - [link](#kix.cw3rkd8chvqy)

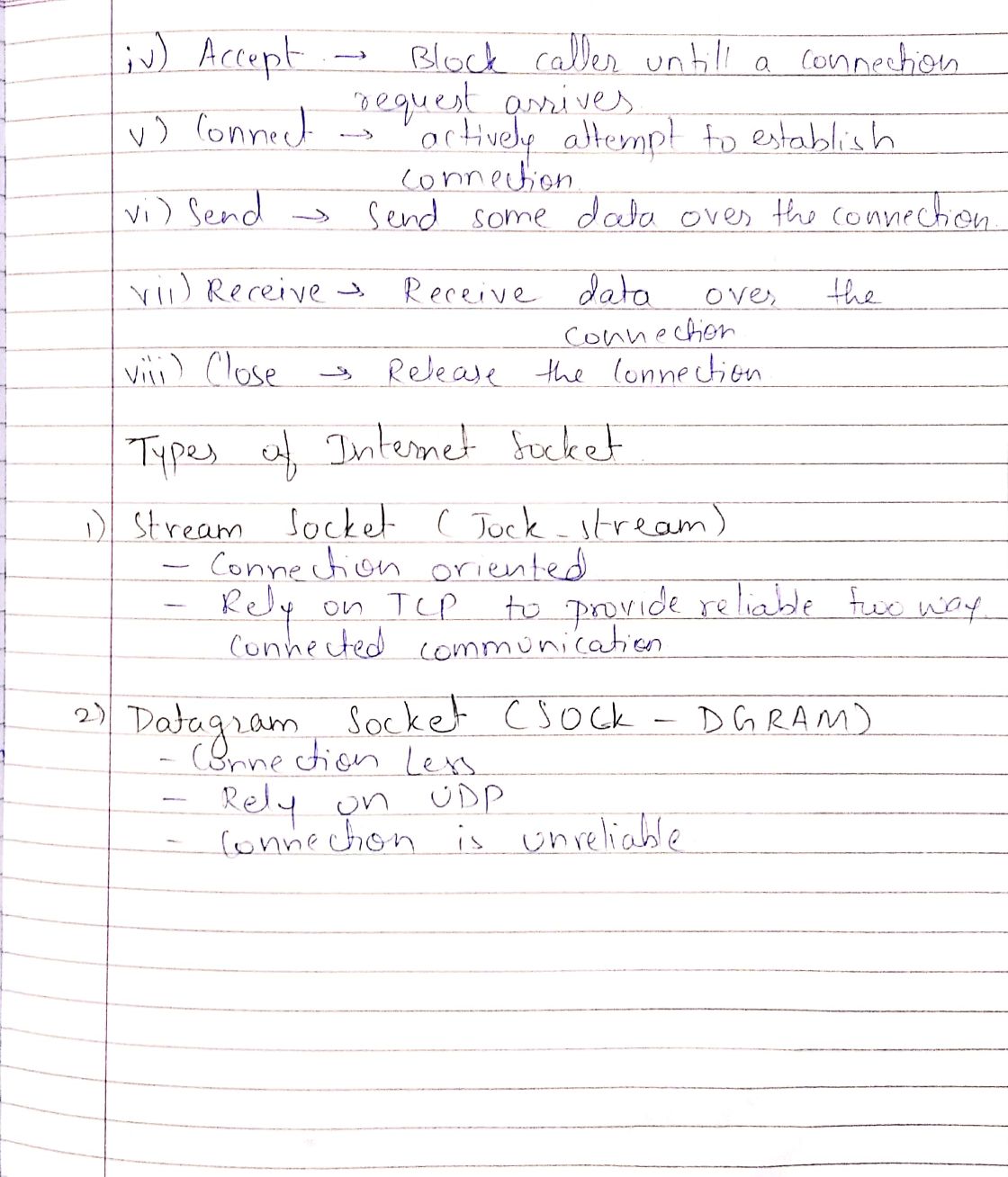
Practical 11 : Design and development of a Distributed application using EJB - [link](#kix.2suqdu90kmdo)

Practical 12 : Case Study Google as a Distributed system. - [link](#kix.md74n3esxyj5)

## horizontal line

**Practical 1 : Design and development of a Distributed application using socket.**

****

****

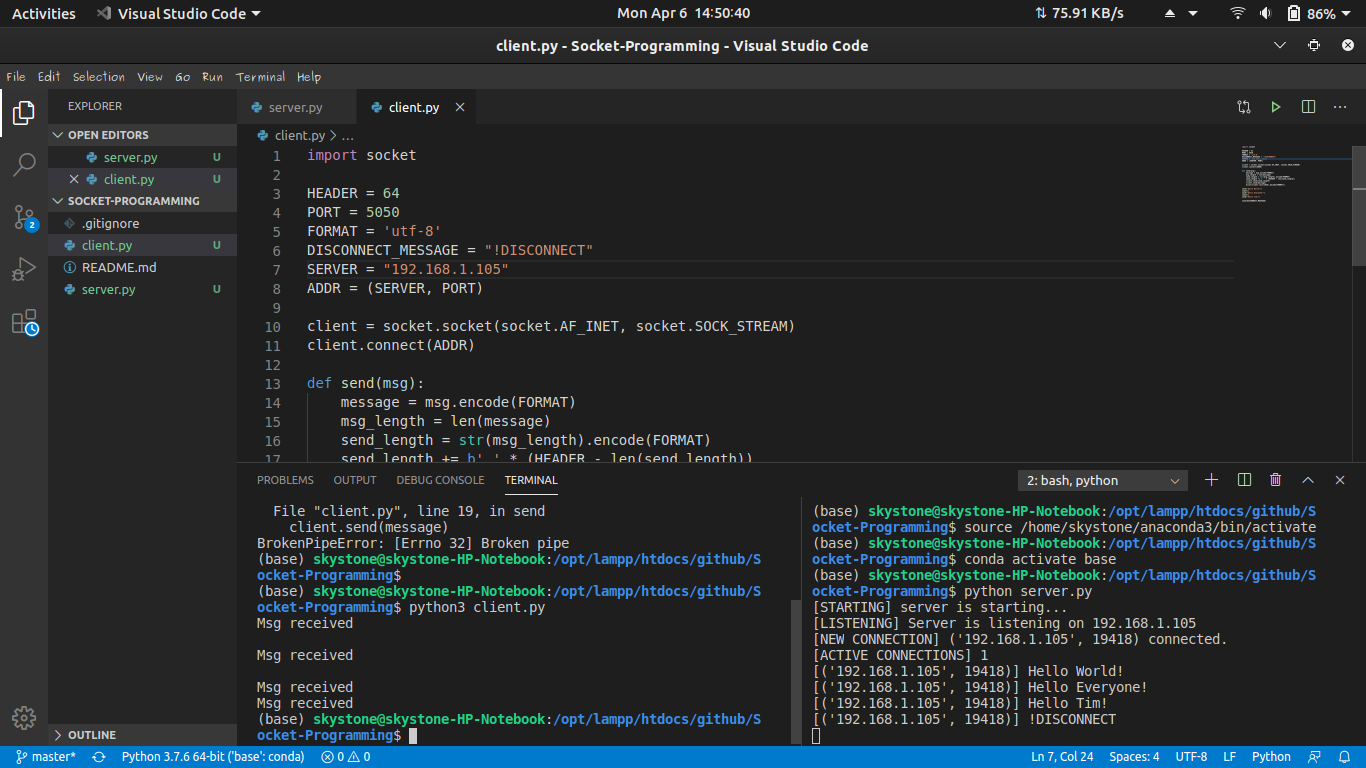
**Program : Server.py**

|  |
| --- |
| import socket import threading  HEADER = 64 PORT = 5050 SERVER = "192.168.1.105" #SERVER = socket.gethostbyname(socket.gethostname()) ADDR = (SERVER, PORT) FORMAT = 'utf-8' DISCONNECT\_MESSAGE = "!DISCONNECT"  server = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) server.bind(ADDR)  def handle\_client(conn, addr):  print(f"[NEW CONNECTION] {addr} connected.")  connected = True  while connected:  msg\_length = conn.recv(HEADER).decode(FORMAT)  if msg\_length:  msg\_length = int(msg\_length)  msg = conn.recv(msg\_length).decode(FORMAT)  if msg == DISCONNECT\_MESSAGE:  connected = False  print(f"[{addr}] {msg}")  conn.send("Msg received".encode(FORMAT))  conn.close()    def start():  server.listen()  print(f"[LISTENING] Server is listening on {SERVER}")  while True:  conn, addr = server.accept()  thread = threading.Thread(target=handle\_client, args=(conn, addr))  thread.start()  print(f"[ACTIVE CONNECTIONS] {threading.activeCount() - 1}")   print("[STARTING] server is starting...") start() |

**Program : Client.py**

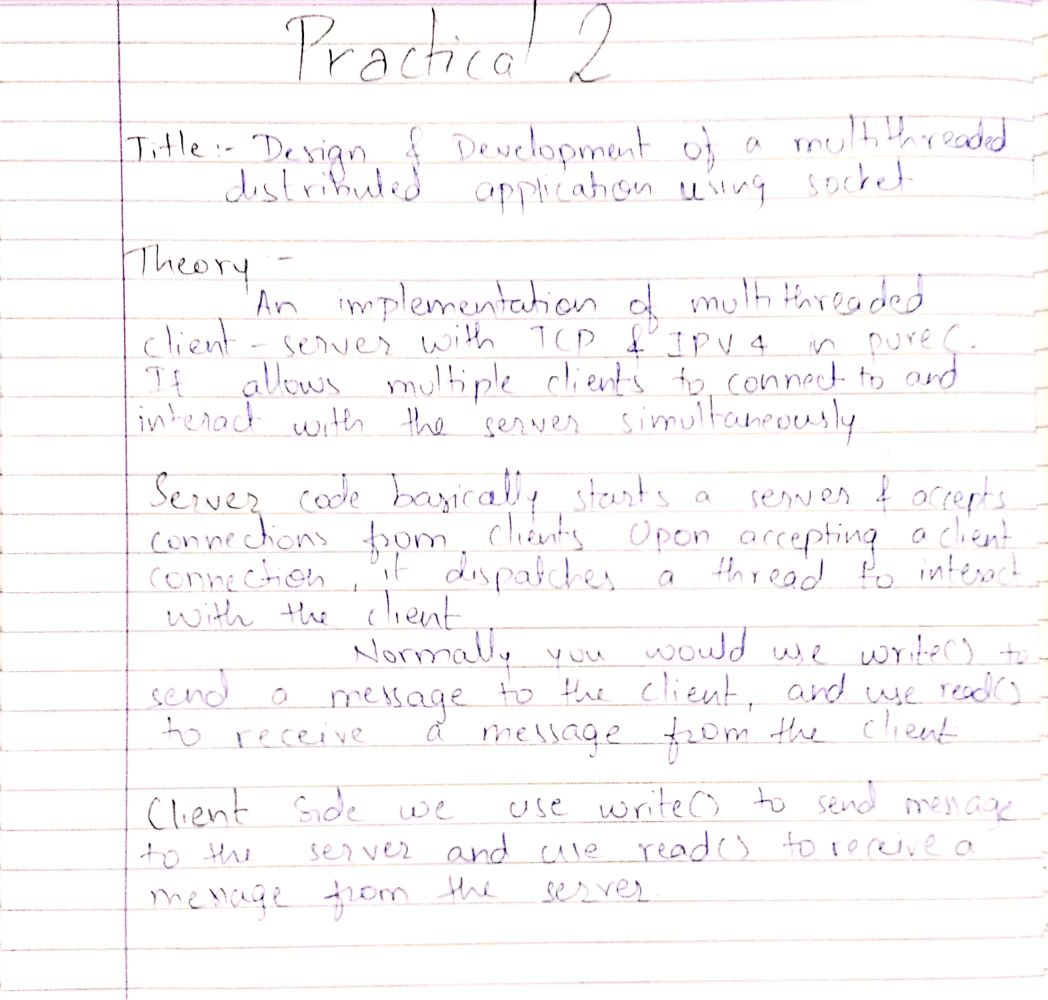
|  |
| --- |
| import socket  HEADER = 64 PORT = 5050 FORMAT = 'utf-8' DISCONNECT\_MESSAGE = "!DISCONNECT" SERVER = "192.168.1.105" ADDR = (SERVER, PORT)  client = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) client.connect(ADDR)  def send(msg):  message = msg.encode(FORMAT)  msg\_length = len(message)  send\_length = str(msg\_length).encode(FORMAT)  send\_length += b' ' \* (HEADER - len(send\_length))  client.send(send\_length)  client.send(message)  print(client.recv(2048).decode(FORMAT))  send("Hello World!") input() send("Hello Everyone!") input() send("Hello Tim!")  send(DISCONNECT\_MESSAGE) |

**Output :**



## horizontal line

**Practical 2 : Design and development of a threaded Distributed application using socket.**

****

**Program : Client.py**

|  |
| --- |
| import socket def Main():  s = socket.socket(socket.AF\_INET,socket.SOCK\_STREAM) s.connect((socket.gethostname() ,1024))  while True:  num1 =input("Enter first numbers:")  s.send(num1.encode('ascii'))  num2 =input("Enter second numbers:")  s.send(num2.encode('ascii'))  data = s.recv(1024)  print('\n',str(data.decode('ascii')))  ans = input('\nDo you want to continue operation with other inputs(y/n) :')  if ans == 'y':  continue  else:  break  s.close() if \_\_name\_\_ == '\_\_main\_\_':  Main() |

**Program : Server.py**

|  |
| --- |
| import socket from \_thread import \* import threading print\_lock = threading.Lock()  def threaded(c):  while True:  num1 = c.recv(1024)  num2 = c.recv(1024)  if not (num1 or num2):  print\_lock.release()  break  add=int(num1)+int(num2)  sub=int(num1)-int(num2)  mul=int(num1)\*int(num2)  div=int(num1)/int(num2)  mod=int(num1)%int(num2)  exp=int(num1)\*\*int(num2)  c.send(("addition: "+str(add)+"\nSubstraction:  "+str(sub)+"\nMultiplication: "+str(mul)+"\nDivision: "+str(div)+"\nModulus: "+str(mod)+"\nExponent: "+str(exp)).encode())  c.close()  def Main():  s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) s.bind((socket.gethostname(),1024))  s.listen(5)  while True:  c, addr = s.accept()  print\_lock.acquire()  print('Connected to :', addr[0], ':', addr[1])  start\_new\_thread(threaded, (c,))  s.close()   if \_\_name\_\_ == '\_\_main\_\_':  Main() |

**Output :**

**skystone@skystone-HP-Notebook**:**~/Desktop/DC Prac/Practical 2**$ python3 server.py

Connected to : 127\_0\_0\_1 : 60012

**skystone@skystone-HP-Notebook**:**~/Desktop/DC Prac/Practical 2**$ python3 client.py

Enter first numbers:5

Enter second numbers:9

Addition: 14

Substraction: -4

Multiplication: 45

Division: 0.5555555555555556

Modulus: 5

Exponent: 1953125

Do you want to continue operation with other inputs(y/n) : y

Enter first numbers:3

Enter second numbers:4

Addition: 7

Substraction: -1

Multiplication: 12

Division: 0.75

Modulus: 3

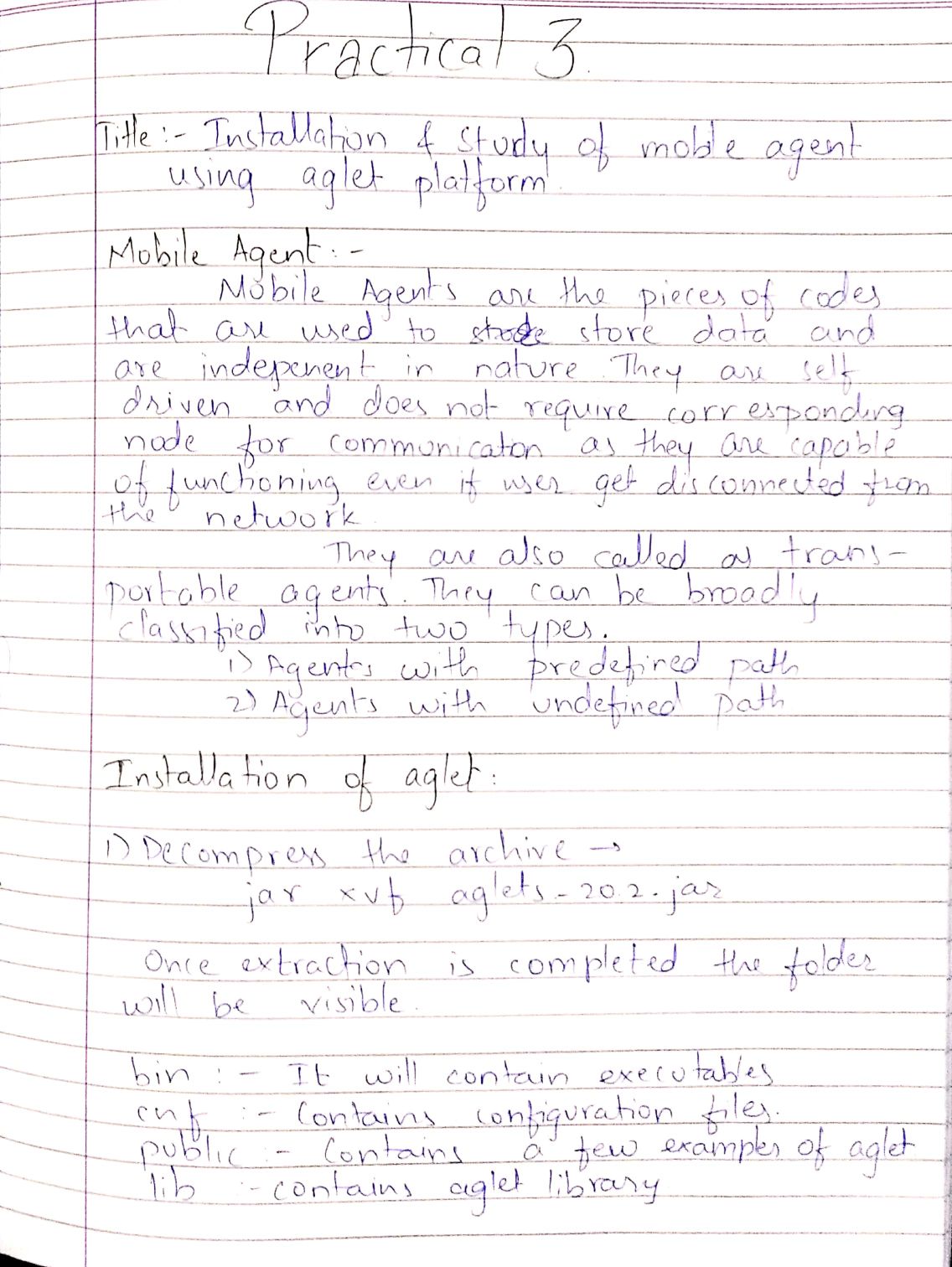
Exponent: 81

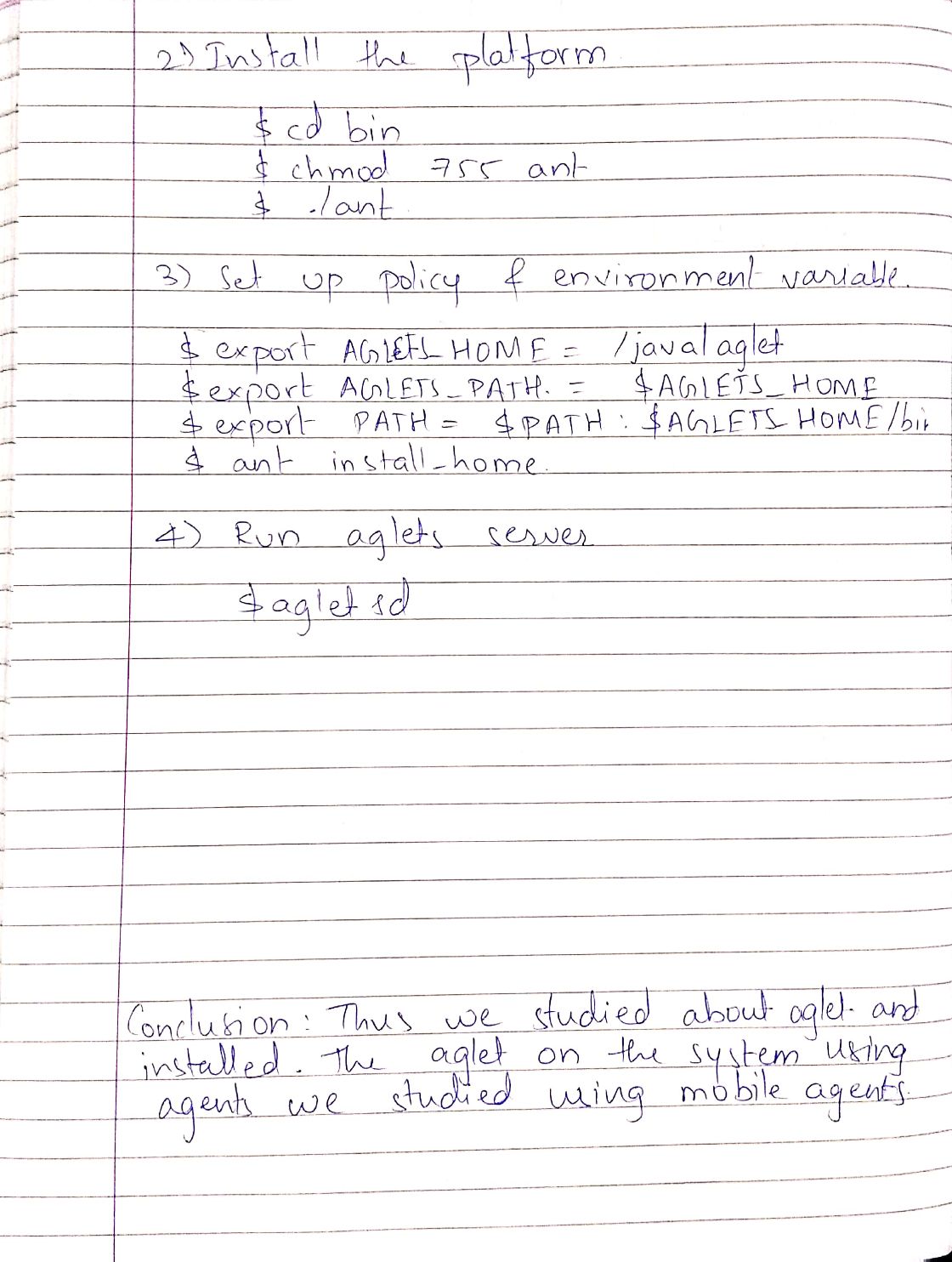
Do you want to continue operation with other inputs(y/n) :n

**skystone@skystone-HP-Notebook**:**~/Desktop/DC Prac/Practical 2**$

horizontal line

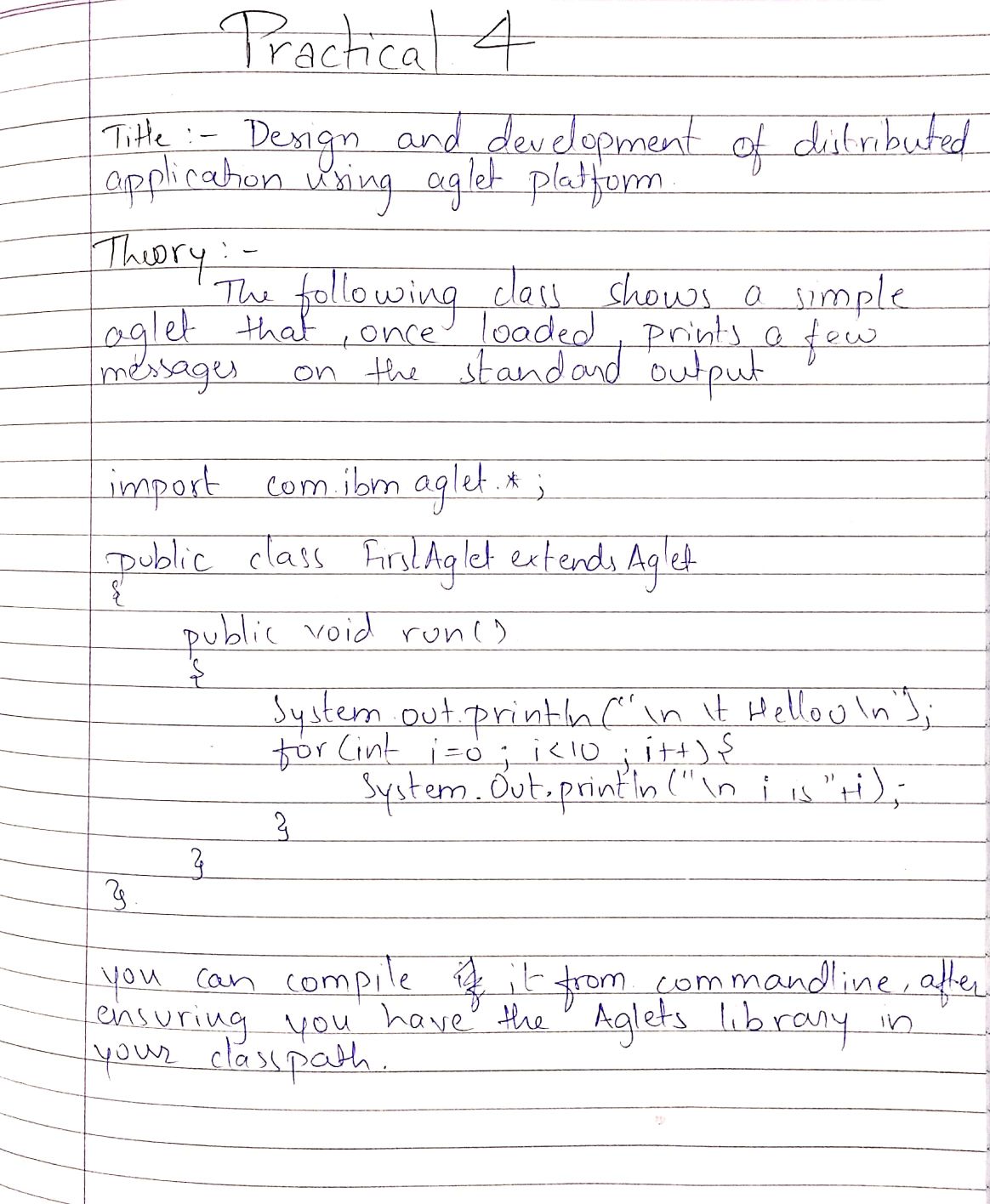
**Practical 3 : Installation and study of Mobile agent using Aglet platform.**

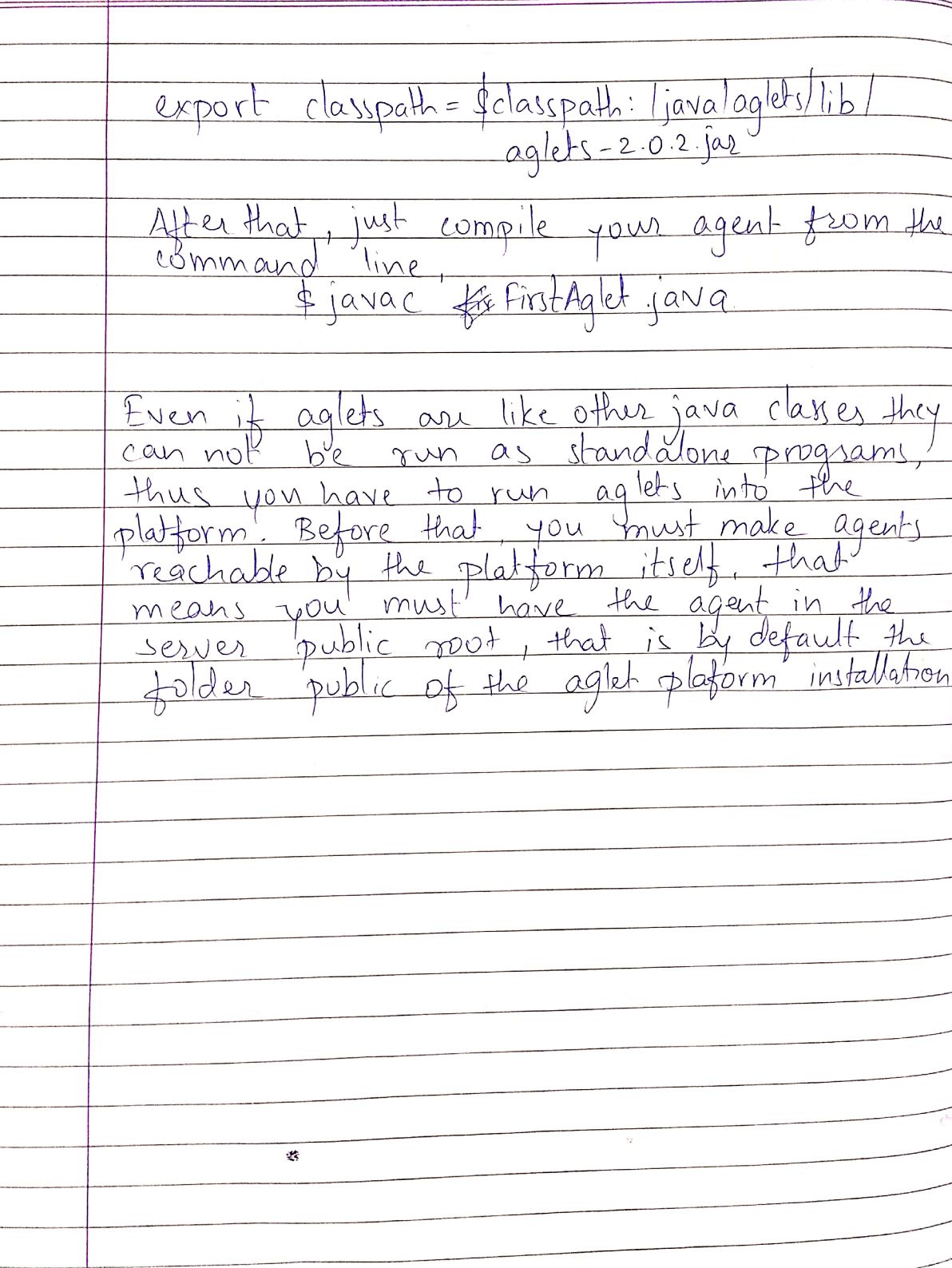
****

****

horizontal line

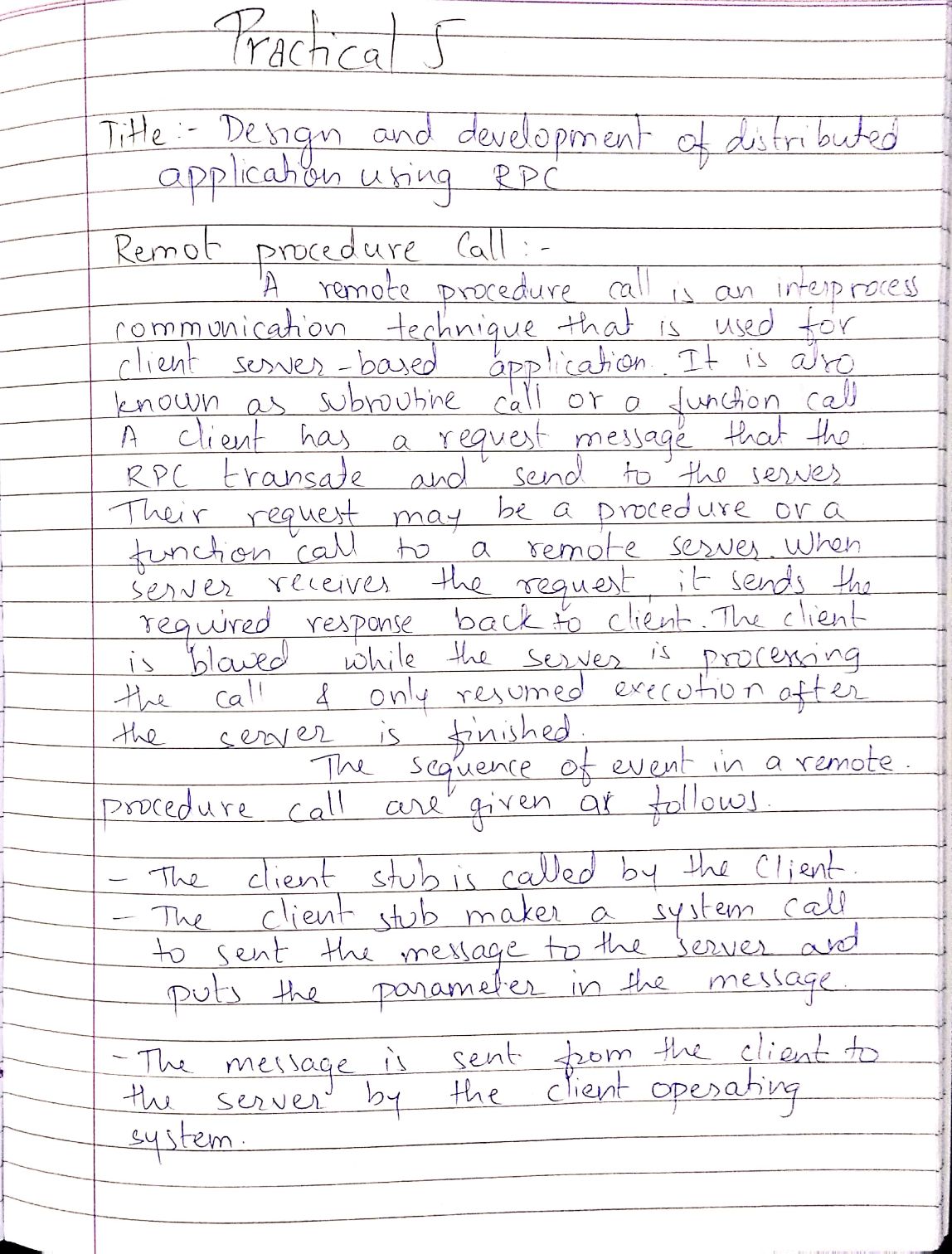
**Practical 4 : Design and development of a Distributed application using Agelt platform.**

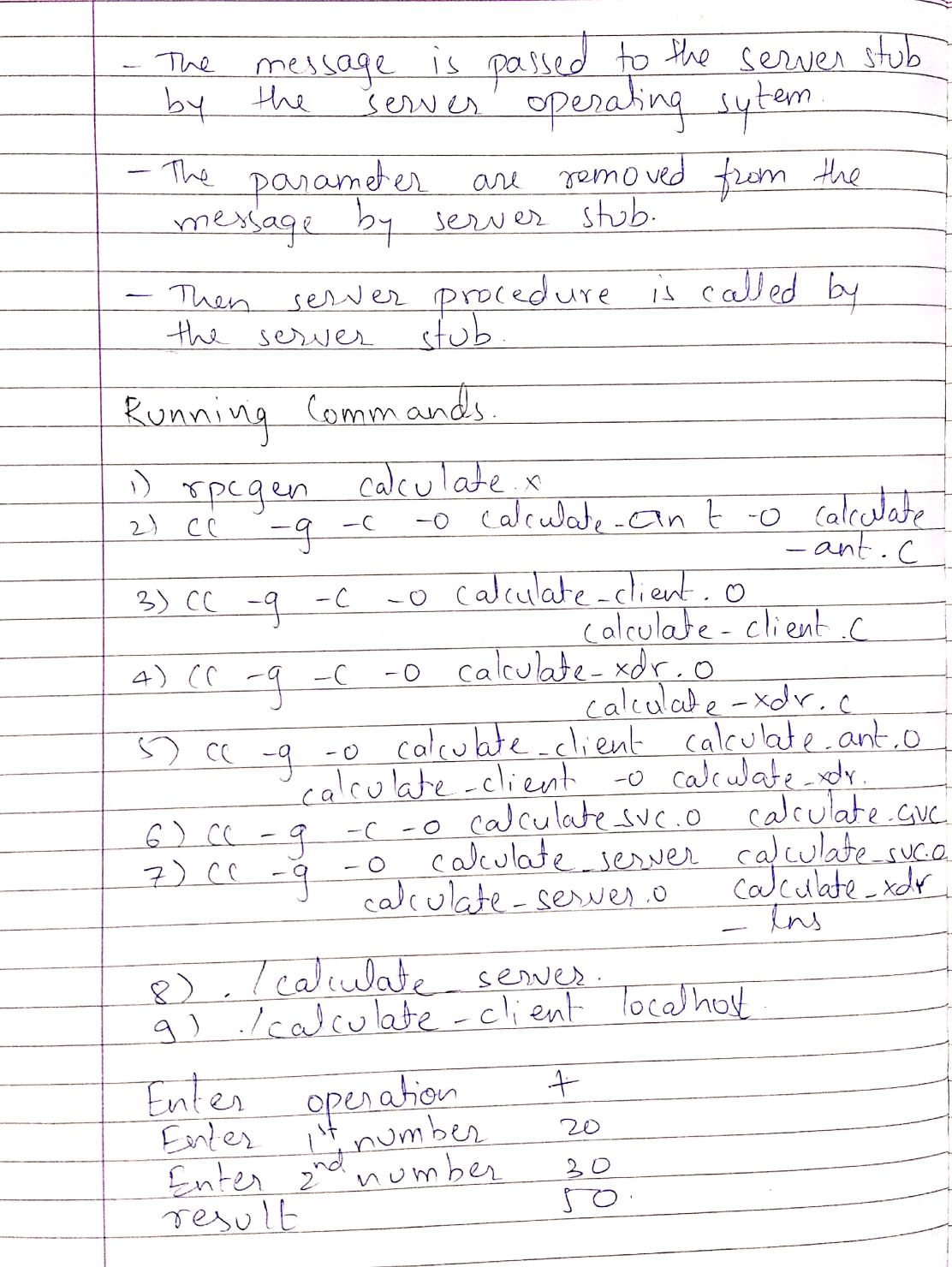
****

****

horizontal line

**Practical 5 : Design and development of a Distributed application using RPC.**

****

****

**Program : ArCompt.java**

|  |
| --- |
| import java.rmi.\*; import java.util.\*; import java.lang.\*; // Creating an Interface public interface ArCompt extends java.rmi.Remote {  // Declaring the method  public int factorial(int x) throws java.rmi.RemoteException;  public double calfun(double x, int n) throws java.rmi.RemoteException; } |

**Program : ArComptImpl.java**

|  |
| --- |
| import java.util.Scanner; import java.rmi.\*; import java.rmi.server.\*; import java.util.\*; import java.lang.\*; public class ArComptImpl extends java.rmi.server.UnicastRemoteObject implements ArCompt {  public ArComptImpl()  throws java.rmi.RemoteException {  super();  }  public int factorial(int x)  throws java.rmi.RemoteException {  if (x == 0) {  return 1;  } else {  return x \* factorial(x - 1);  }  }  public double calfun(double x1, int y) throws java.rmi.RemoteException {  double radians = Math.toRadians(x1);  if (y == 1) {  return Math.sin(radians);  } else if (y == 2) {  return Math.cos(radians);  } else if (y == 3) {  return Math.tan(radians);  } else {  return 0;  }  } } |

**Program : ArComptClient.java**

|  |
| --- |
| import java.rmi.\*; import java.rmi.server.\*; import java.util.\*; import java.io.\*; import java.lang.\*; import java.net.\*; public class ArComptClient {  public static void main(String[] args) {  Scanner s = new Scanner(System.in);  try {  ArCompt c = (ArCompt) Naming.lookup("rmi://localhost/ArCompt");  System.out.print("\nEnter the Number to find factorial:");  int num = s.nextInt();  System.out.print("\nFactorial: " + c.factorial(num));  System.out.print("\nEnter the value (in degree) to find sin():");  double a = s.nextDouble();  System.out.print("\nSin(): " + c.calfun(a, 1));  System.out.print("\nEnter the value (in degree) to find cos():");  double b = s.nextDouble();  System.out.print("\ncos(): " + c.calfun(b, 2));  System.out.print("\nEnter the value (in degree) to find tan():");  double t = s.nextDouble();  System.out.print("\ntan(): " + c.calfun(t, 3));  } catch (Exception e) {  System.out.print(e);  }  } } |

**Program : ArComptServer.java**

|  |
| --- |
| import java.rmi.\*; import java.rmi.server.\*; public class ArComptServer {  public static void main(String[] args) {  try {  ArComptImpl stub = new ArComptImpl();  Naming.rebind("rmi:///ArCompt", stub);  } catch (Exception e) {  System.out.print(e);  }  } } |

**Output :**

**skystone@skystone-HP-Notebook**:**~/Desktop/DC Prac/Practical 5**$ javac ArCompt.java

**skystone@skystone-HP-Notebook**:**~/Desktop/DC Prac/Practical 5**$ javac ArComptImpl.java

**skystone@skystone-HP-Notebook**:**~/Desktop/DC Prac/Practical 5**$ javac ArComptServer.java

**skystone@skystone-HP-Notebook**:**~/Desktop/DC Prac/Practical 5**$ javac Arfomptflient.java

**skystone@skystone-HP-Notebook**:**~/Desktop/DC Prac/Practical 5**$ javac ArComptClient.java

**skystone@skystone-HP-Notebook**:**~/Desktop/DC Prac/Practical 5**$ rmic ArComptImpl

Warning: generation and use of skeletons and static stubs for JR. is deprecated. Skeletons are unnecessary. and static stubs have been superseded by dynamically generated stubs. Users are encouraged to migrate amay from using rmic to generate skeletons and static stubs. See the documentation for java.rmi.server.UnicastRemoteObject.

**skystone@skystone-HP-Notebook**:**~/Desktop/DC Prac/Practical 5**$ rmiregistery

**skystone@skystone-HP-Notebook**:**~/Desktop/DC Prac/Practical 5**$ java ArComptServer

**skystone@skystone-HP-Notebook**:**~/Desktop/DC Prac/Practical 5**$ java ArComptClient

Enter the Number to find factorial:3

Factorial: 6

Enter the value (in degree) to find sin():30

Sin(): 0.49999099999999994

Enter the value (in degree) to find cos():60

cos(): 0.5000000000000001

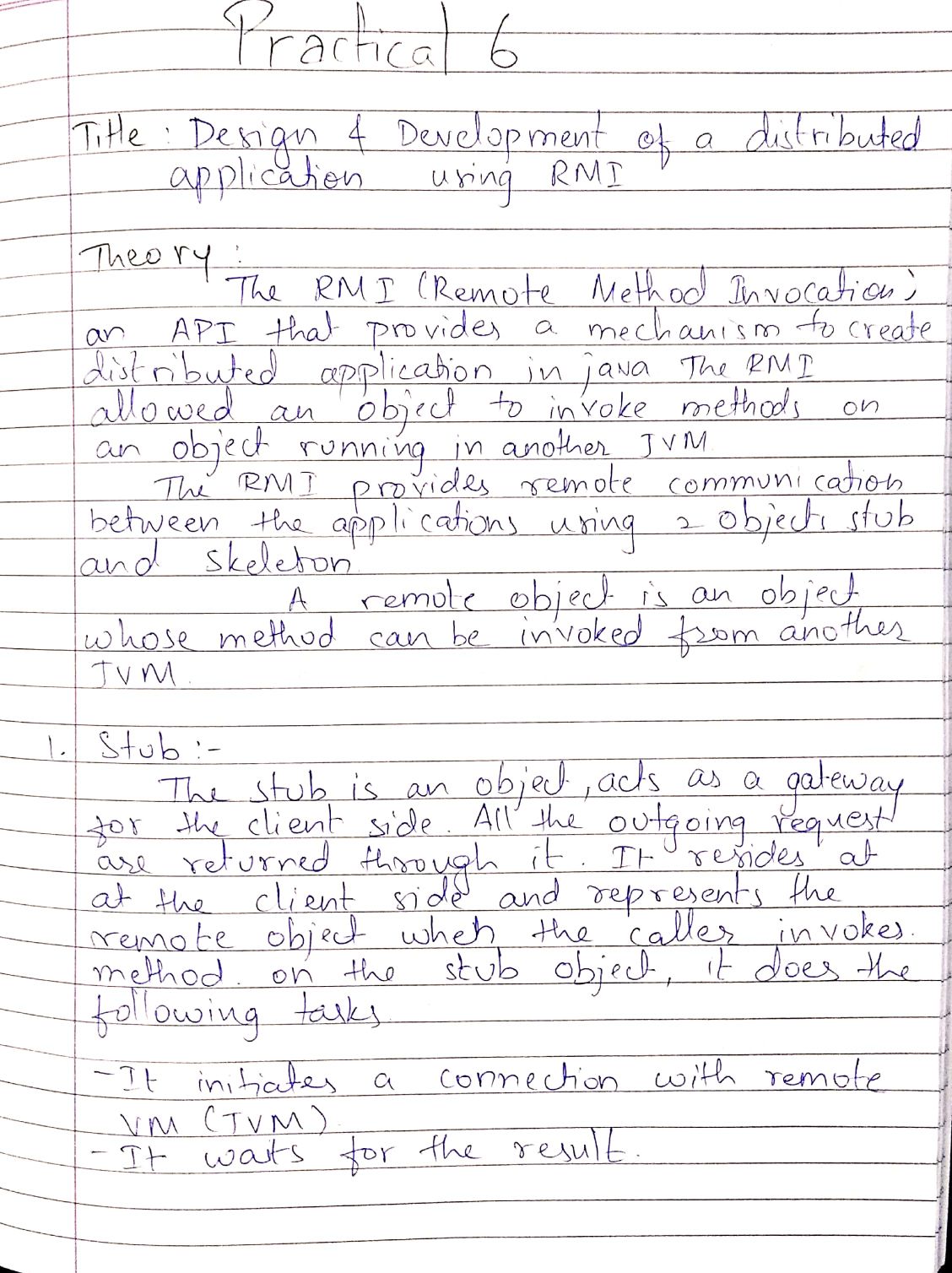
Enter the value (in degree) to find tan():45

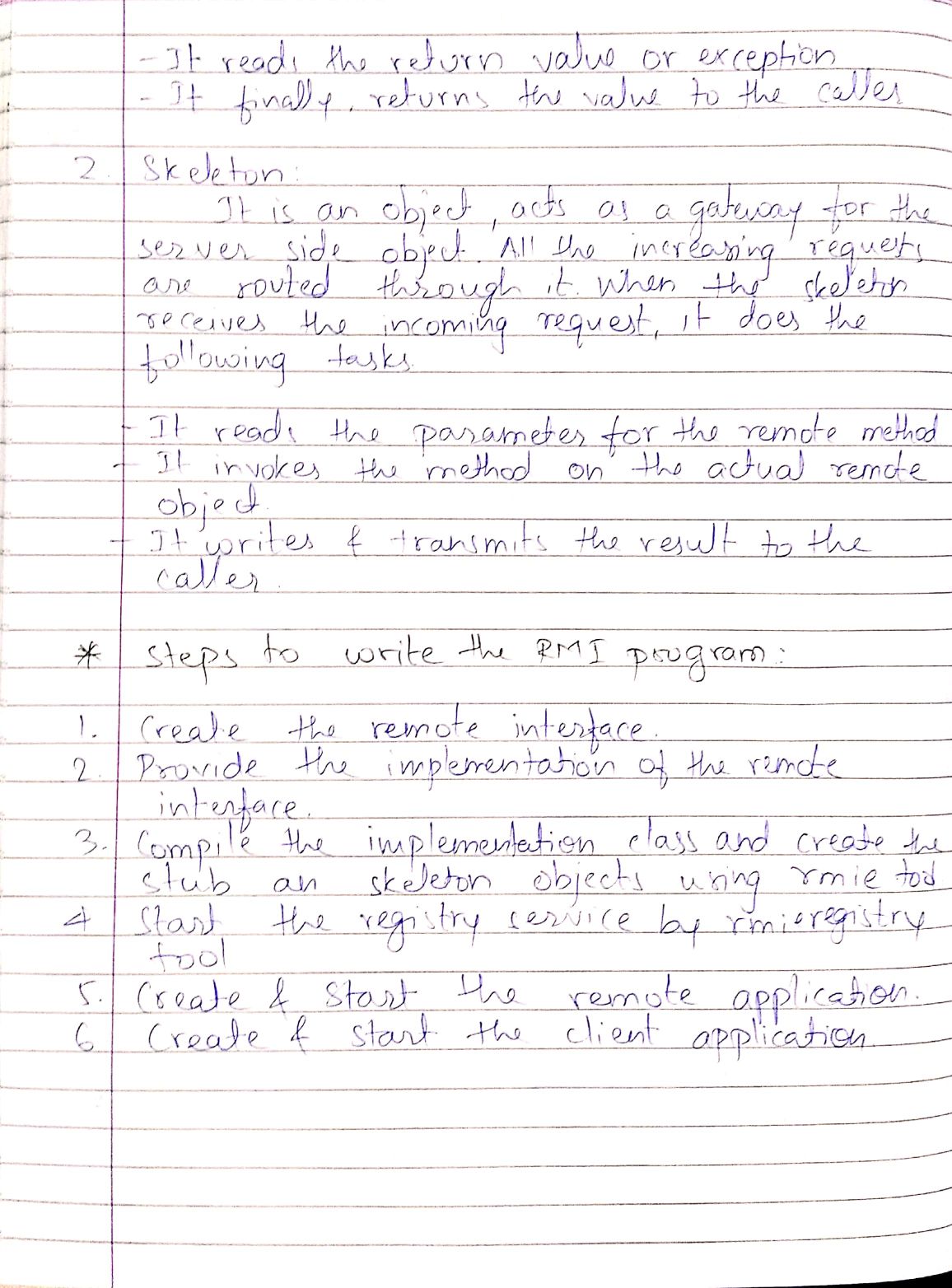
tan(): 0.9999999999999999dell

**skystone@skystone-HP-Notebook**:**~/Desktop/DC Prac/Practical 5**$

horizontal line

**Practical 6 : Design and development of a Distributed application using RMI.**

****

****

**Program:**

**Calculator Interface :**

|  |
| --- |
| import java.rmi.Remote; import java.rmi.RemoteException; public interface CalcInterface extends Remote {  public int add(int x, int y) throws RemoteException;  public int sub(int x, int y) throws RemoteException;  public int mul(int x, int y) throws RemoteException;  public int div(int x, int y) throws RemoteException; } |

**Calculator RMI :**

|  |
| --- |
| import java.rmi.RemoteException; import java.rmi.server.UnicastRemoteObject; public class CalcRmi extends UnicastRemoteObject implements CalcInterface {  public CalcRmi() throws RemoteException {  int a, b;  }  public int add(int a, int b) throws RemoteException {  return a + b;  }  public int sub(int a, int b) throws RemoteException {  return a - b;  }  public int mul(int a, int b) throws RemoteException {  return a \* b;  }  public int div(int a, int b) throws RemoteException {  return a / b;  } } |

**Calculator Server :**

|  |
| --- |
| import java.rmi.registry.Registry; public class CalcServer {  public static void main(String args[]) {  try {  Registry r = java.rmi.registry.LocateRegistry.createRegistry(1099);  r.rebind("Calc", new CalcRmi());  System.out.println("Server connected");  } catch (Exception e) {  System.out.println("Server not connected " + e);  }  } } |

**Calculator Client :**

|  |
| --- |
| import java.rmi.Naming; import java.util.Scanner; public class CalcClient {  public static void main(String args[]) {  System.out.println("MENU");  System.out.println("");  System.out.println("Enter 1 for addition");  System.out.println("Enter 2 for substraction");  System.out.println("Enter 3 for multiplication");  System.out.println("Enter 4 for divition");  System.out.println("");  System.out.println("Enter your choice");  Scanner sc = new Scanner(System.in);  try {  CalcInterface c = (CalcInterface) Naming.lookup("//localhost/Calc");  int choice = sc.nextInt();  int x, y;  switch (choice) {  case 1:  {  System.out.println("Enter the first value");  x = sc.nextInt();  System.out.println("Enter the second value");  y = sc.nextInt();  System.out.println("Answer is : " + c.add(x, y));  Break;  }  case 2:  {  System.out.println("Enter the first value");  x = sc.nextInt();  System.out.println("Enter the second value");  y = sc.nextInt();  System.out.println("Answer is : " + c.sub(x, y));  Break;  }  case 3:  {  System.out.println("Enter the first value");  x = sc.nextInt();  System.out.println("Enter the second value");  y = sc.nextInt();  System.out.println("Answer is : " + c.mul(x, y));  Break;  }  case 4:  {  System.out.println("Enter the first value");  x = sc.nextInt();  System.out.println("Enter the second value");  y = sc.nextInt();  System.out.println("Answer is : " + c.div(x, y));  Break;  }  }  } catch (Exception e) {  System.out.println(e);  }  } } |

**Output :**

**skystone@skystone-HP-Notebook**:**~/Desktop/DC Prac/Practical 6**$ javac CalcInterface.java

Picked up \_JAVAOPTIONS: -DJava.net.preferIPv6Addresses=true

**skystone@skystone-HP-Notebook**:**~/Desktop/DC Prac/Practical 6**$ javac CalcRmi.java

Picked up \_JAVAOPTIONS: -DJava.net.preferIPv6Addresses=true

**skystone@skystone-HP-Notebook**:**~/Desktop/DC Prac/Practical 6**$ javac CalcServer.java

Picked up \_JAVAOPTIONS: -DJava.net.preferIPv6Addresses=true

**skystone@skystone-HP-Notebook**:**~/Desktop/DC Prac/Practical 6**$ javac CalcClient.java

Picked up \_JAVAOPTIONS: -DJava.net.preferIPv6Addresses=true

**skystone@skystone-HP-Notebook**:**~/Desktop/DC Prac/Practical 6**$ java CalcServer

Picked up \_JAVAOPTIONS: -DJava.net.preferIPv6Addresses=true

Server Connected

^C**skystone@skystone-HP-Notebook**:**~/Desktop/DC Prac/Practical 6**$

**skystone@skystone-HP-Notebook**:**~/Desktop/DC Prac/Practical 6**$ Java CalcClient

Picked up JAVA OPTIONS: -Djava.net.preferIPv6Addresses=true

MENU

Enter 1 for addition

Enter 2 for substraction

Enter 3 for multiplication

Enter 4 for division

Enter your choice: 2

Enter the first value: 12

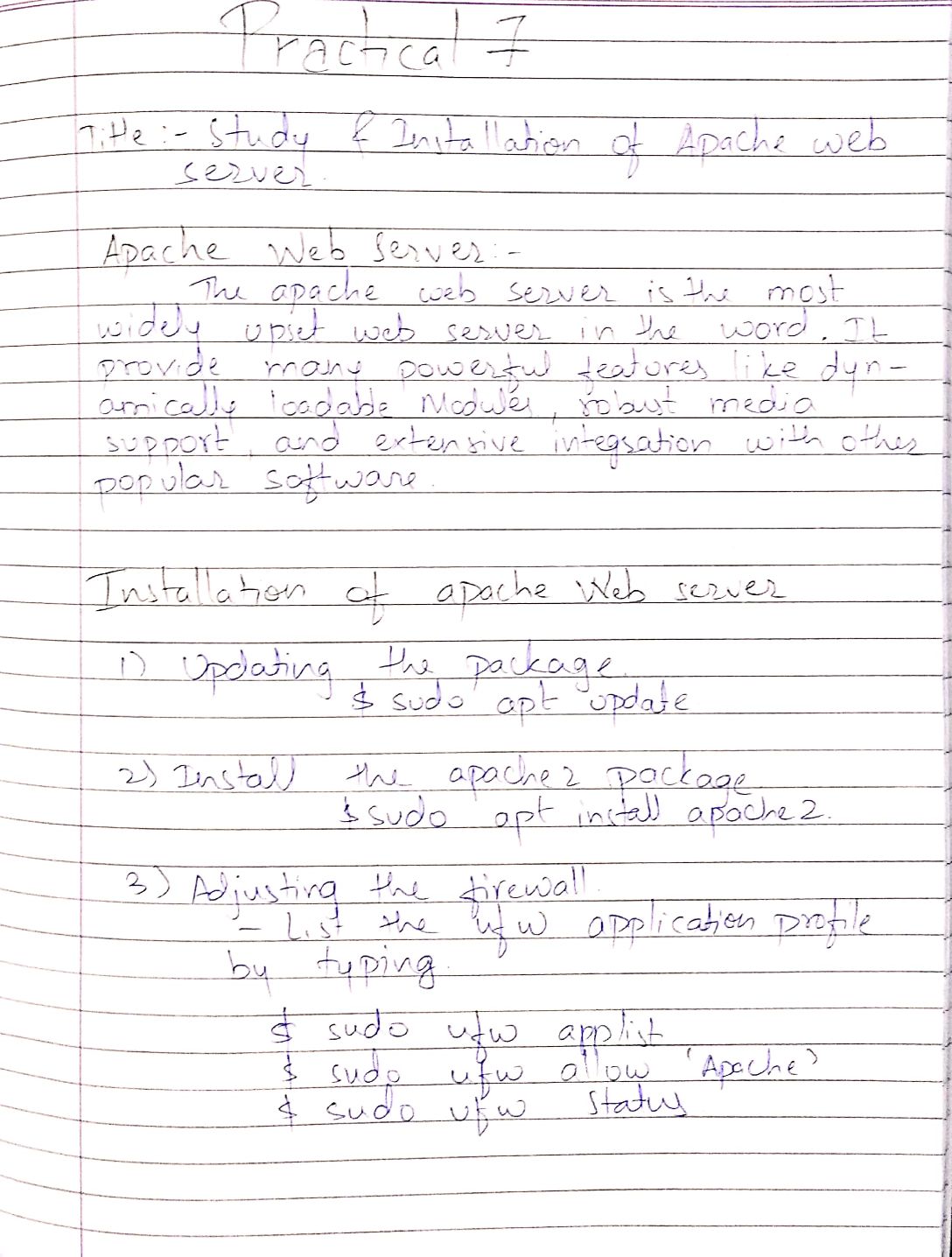
Enter the second value: 21

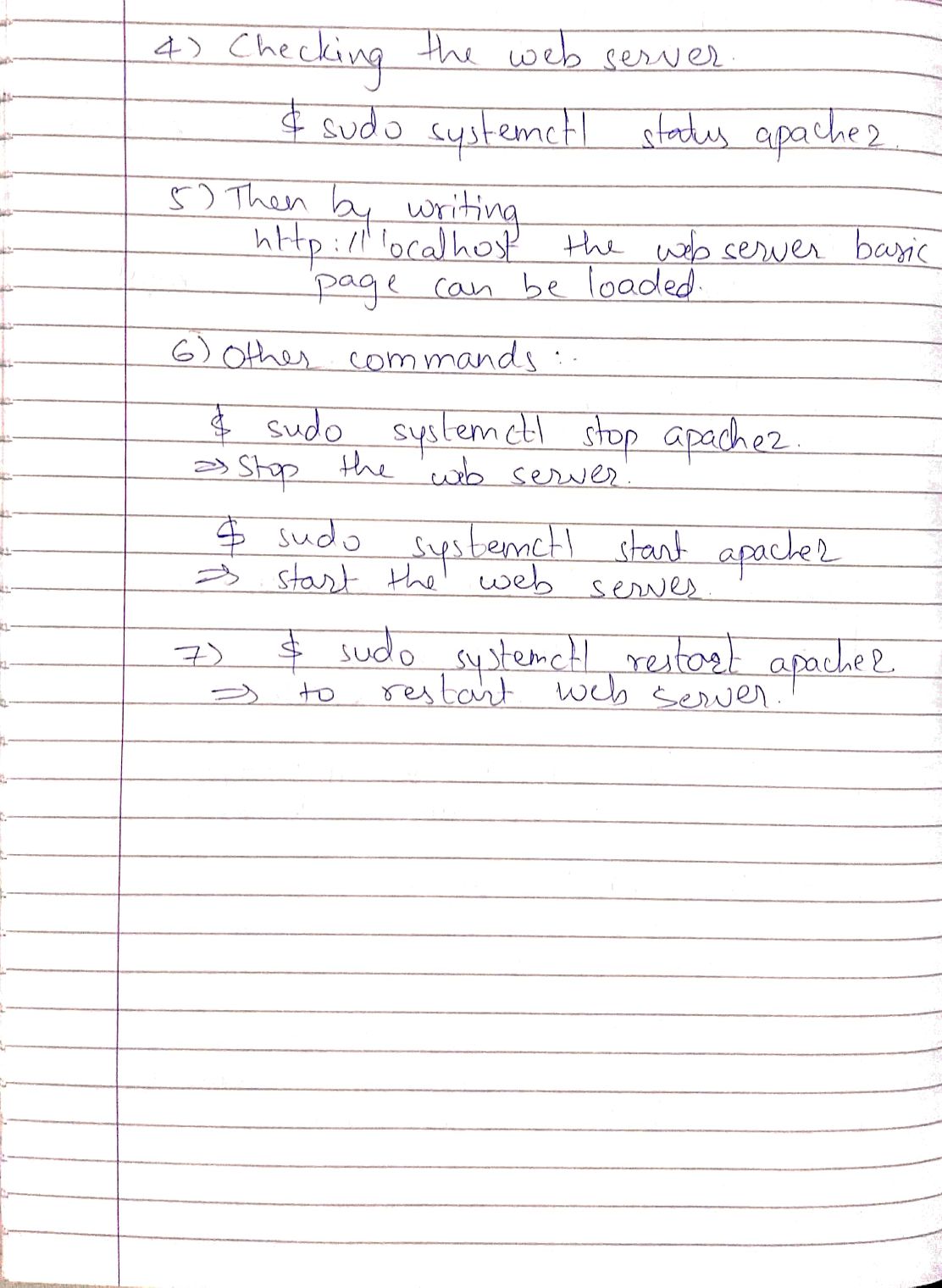
Answer is : -9

**skystone@skystone-HP-Notebook**:**~/Desktop/DC Prac/Practical 6**$

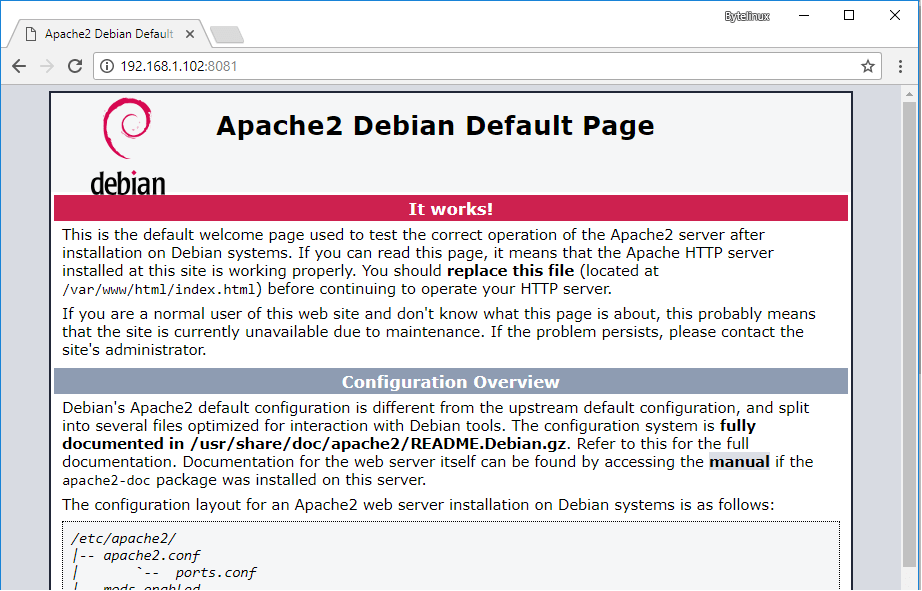
horizontal line

**Practical 7 : Study and Installation of Apache Web Server.**

****

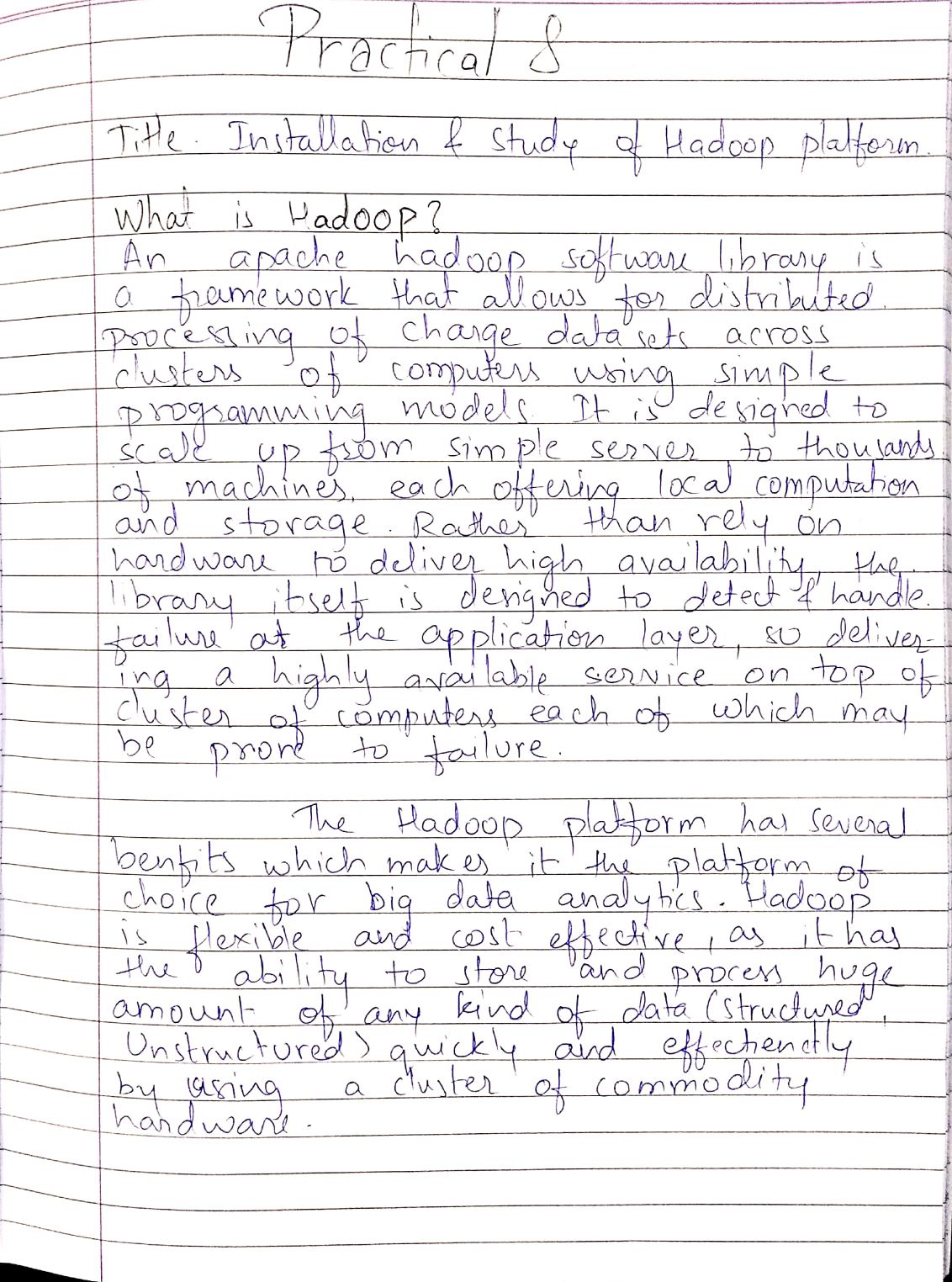
****

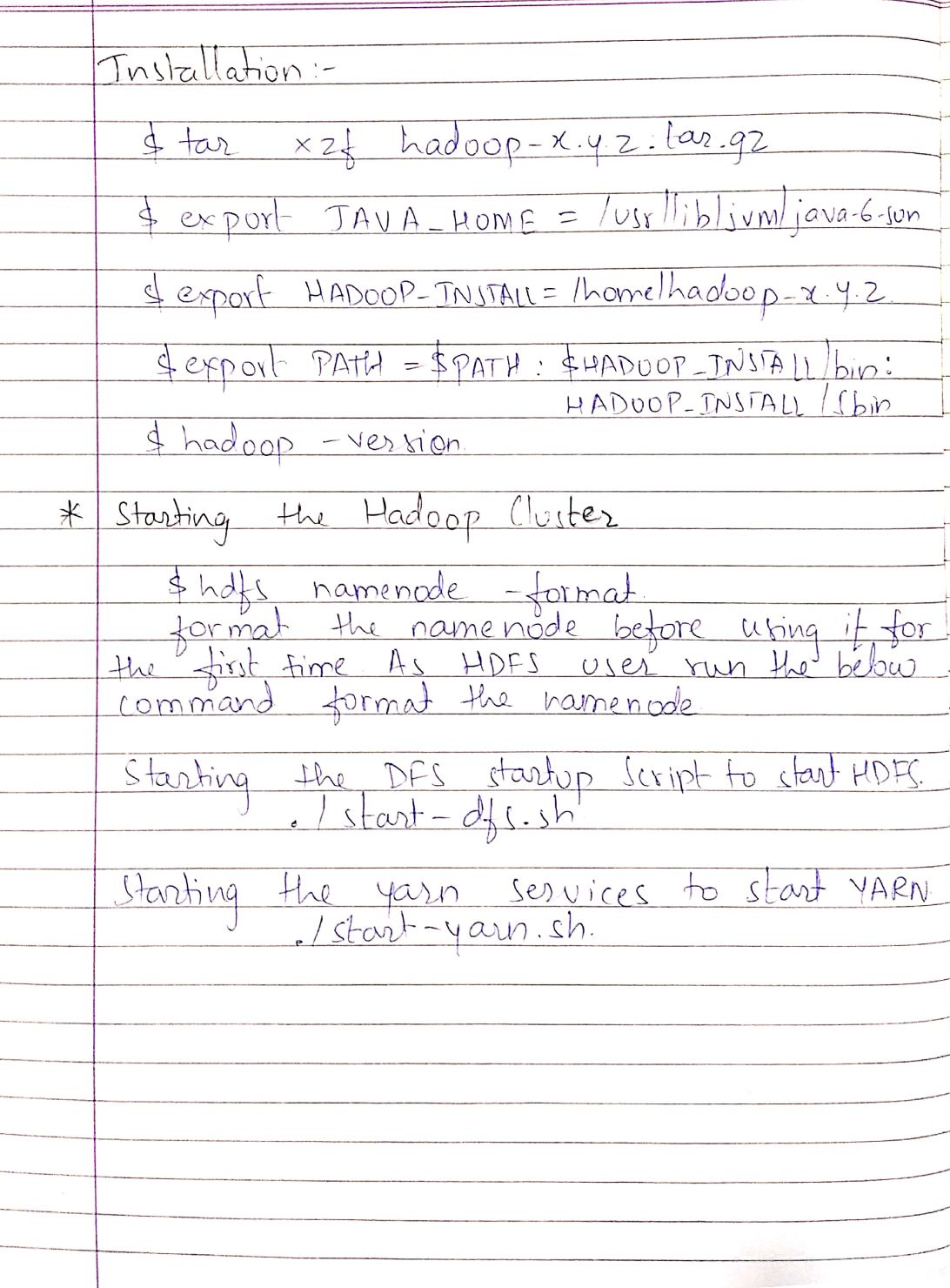
**OUTPUT :**



horizontal line

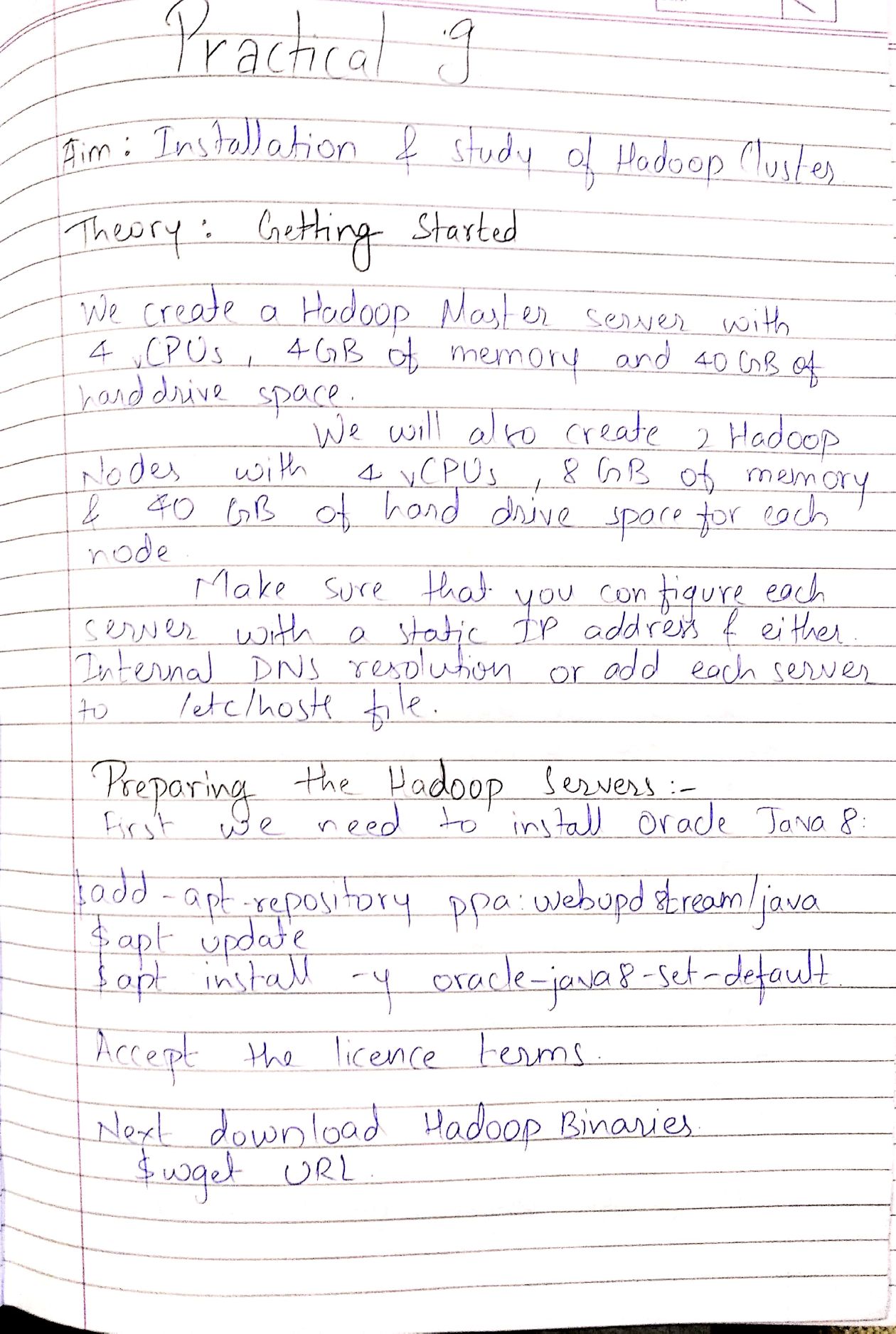
**Practical 8 : Installation and study of Hadoop Platform.**

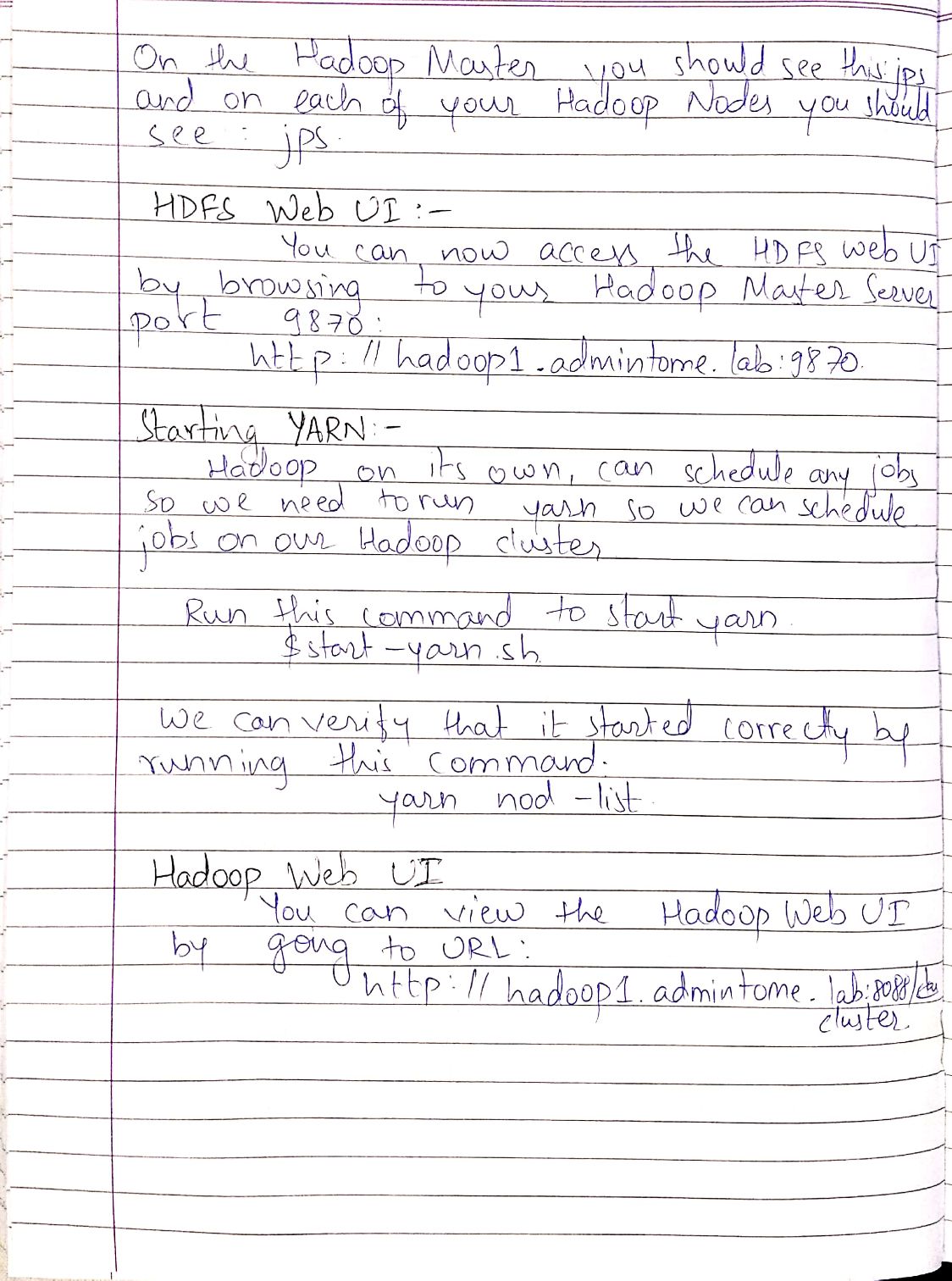
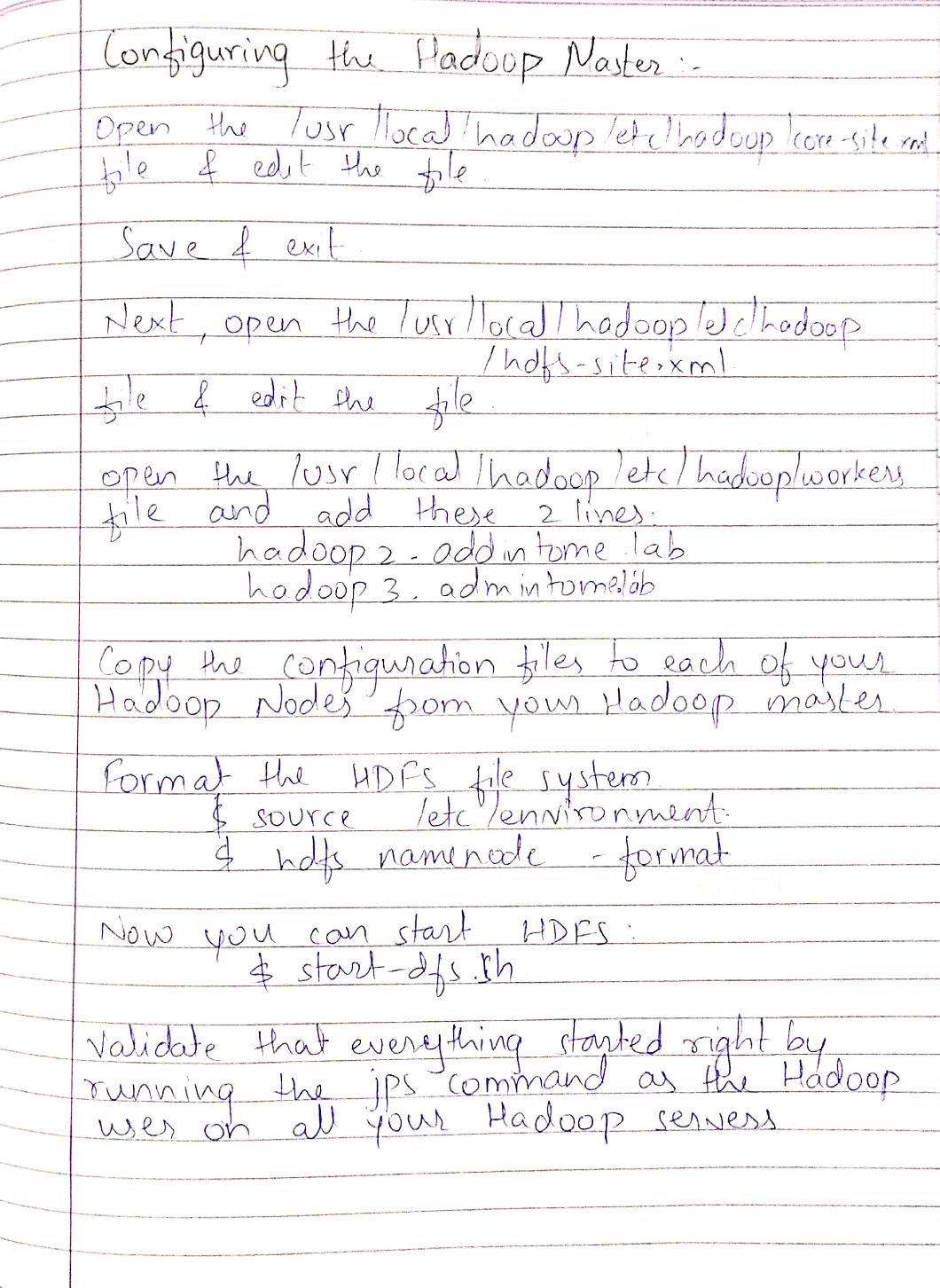
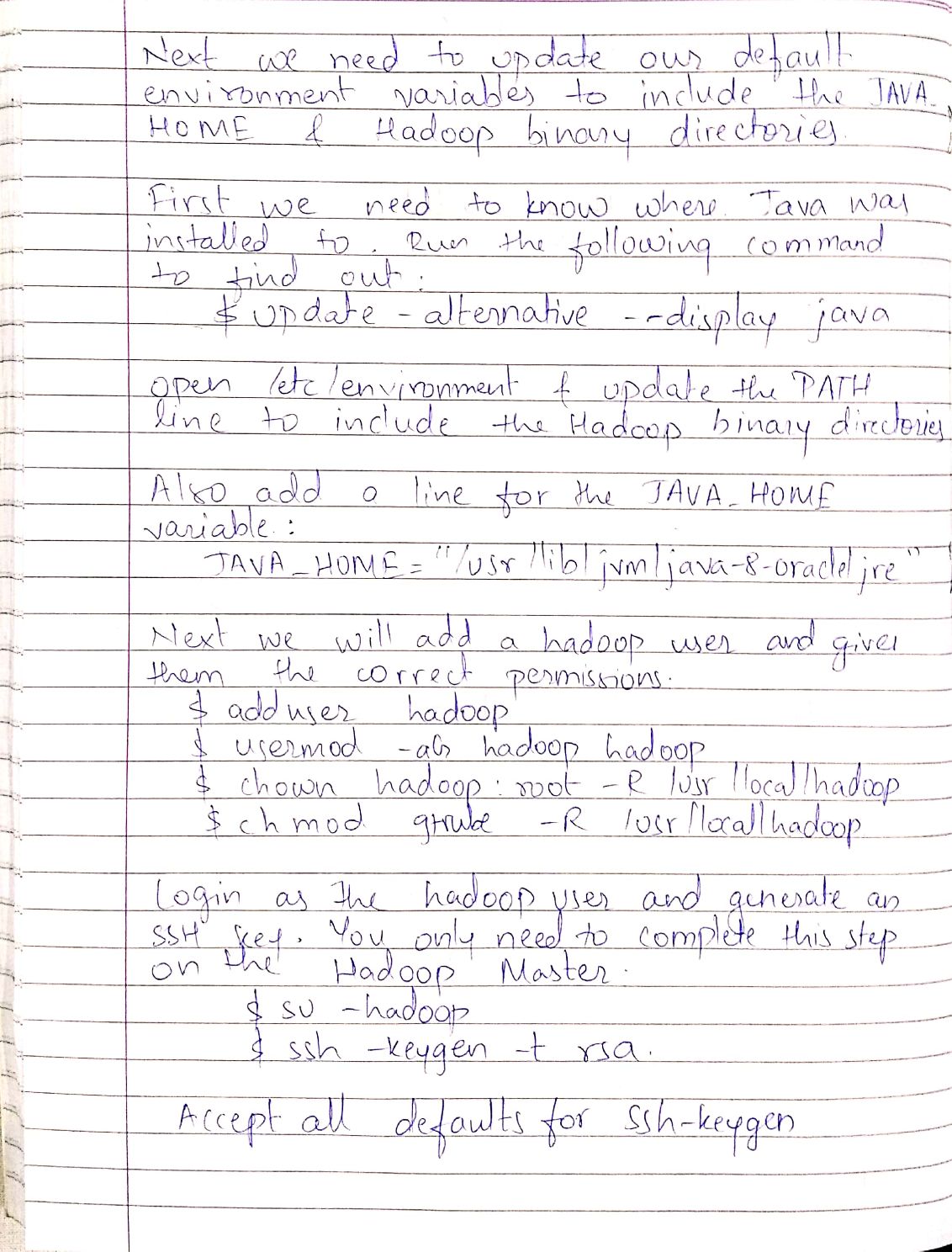
****

****

horizontal line

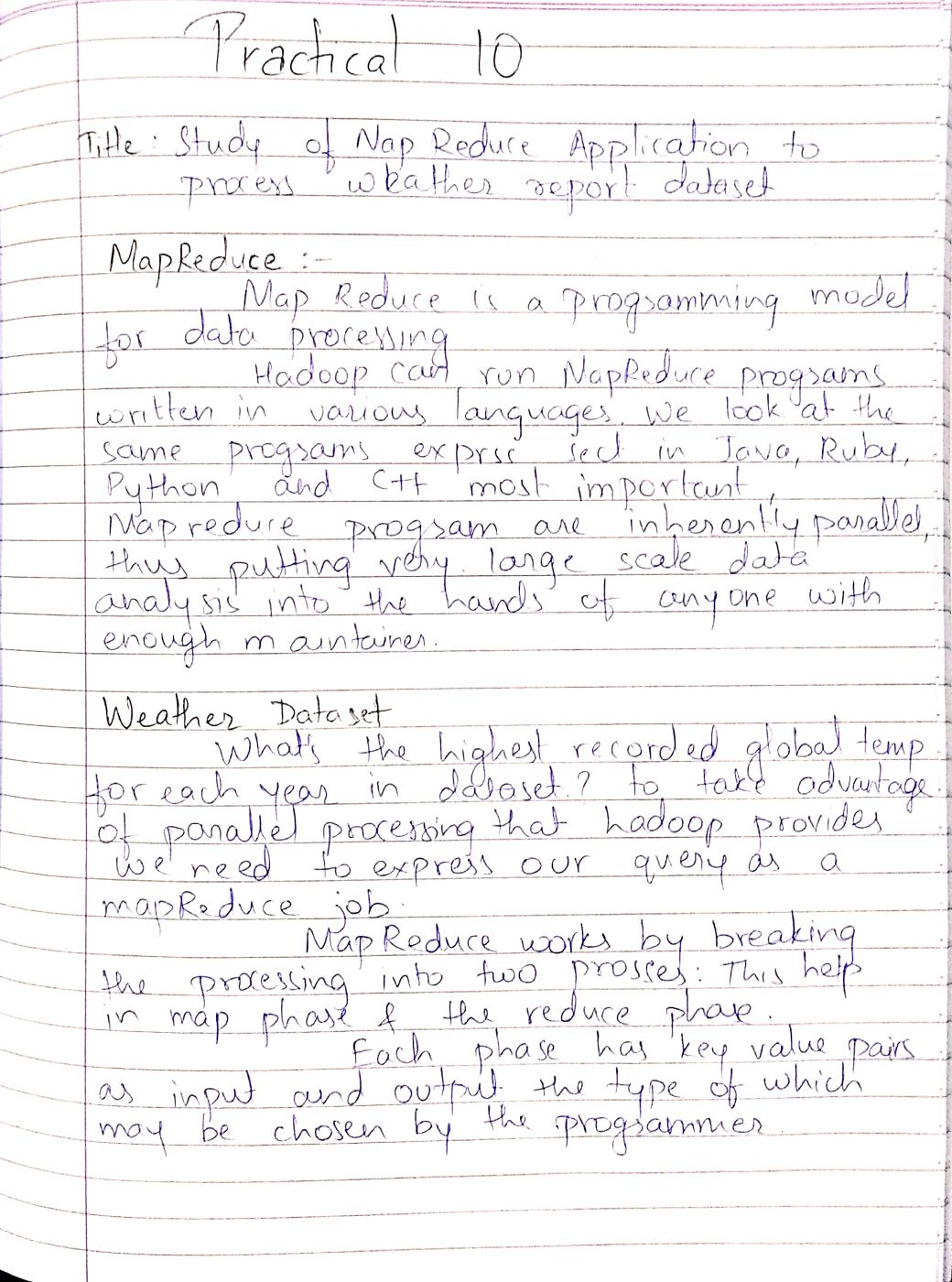
**Practical 9 : Installation and study of Hadoop Cluster.**

****



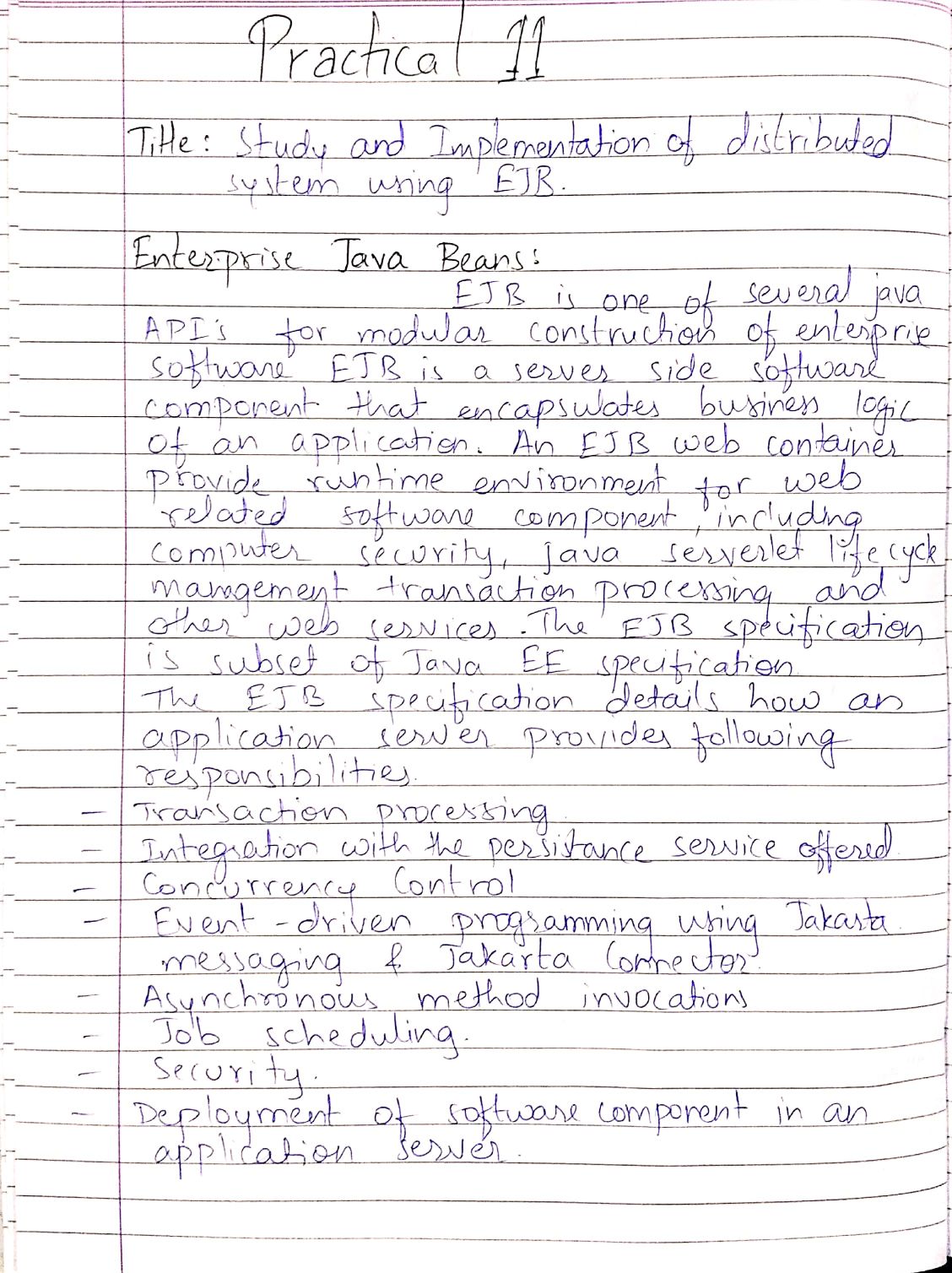
horizontal line

**Practical 10 : Study of MapReduce Application to process weather report dataset.**

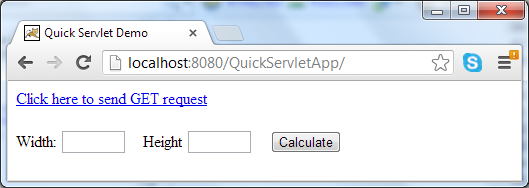
****

horizontal line

**Practical 11 : Design and development of a Distributed application using EJB**

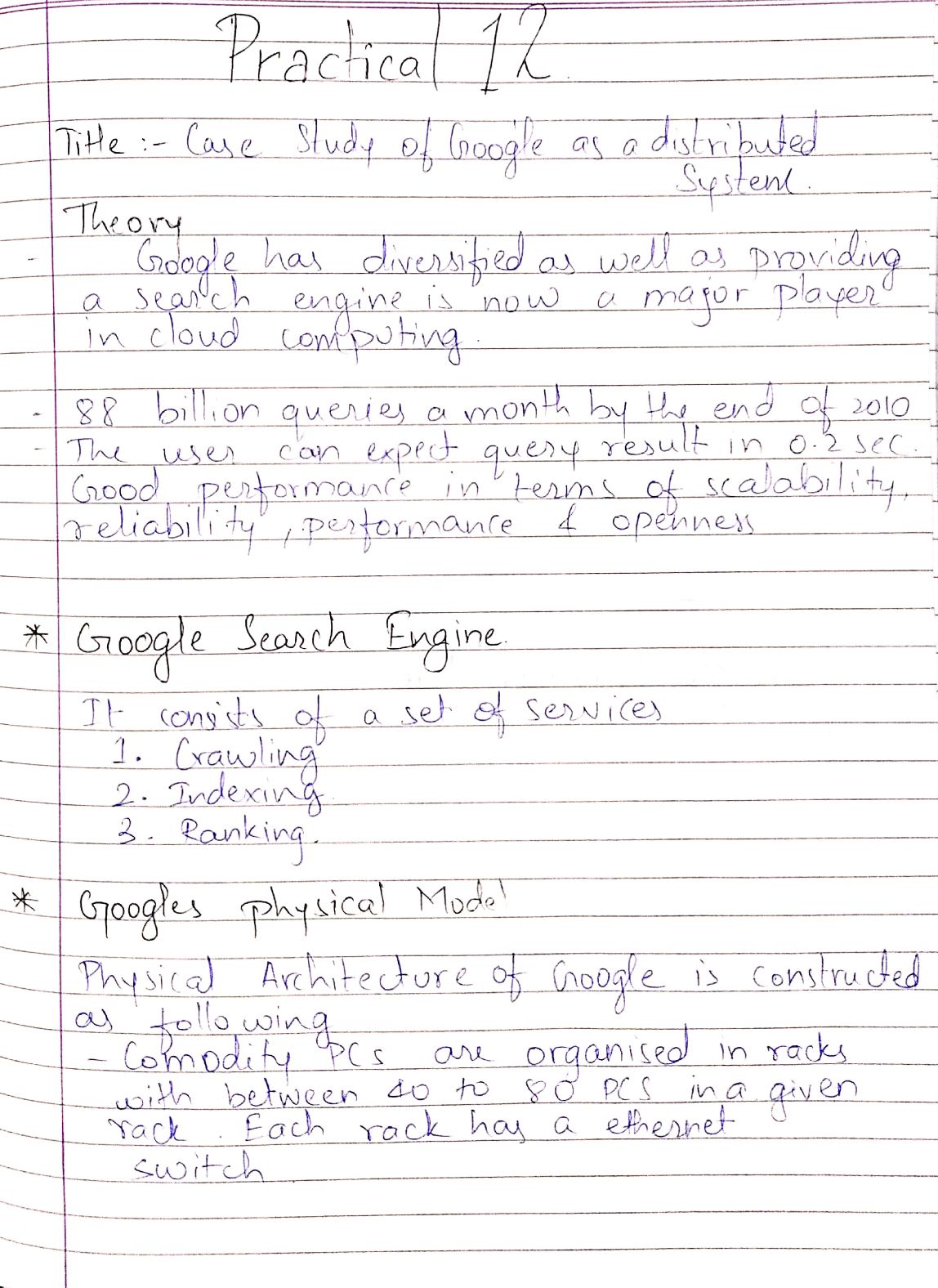
****

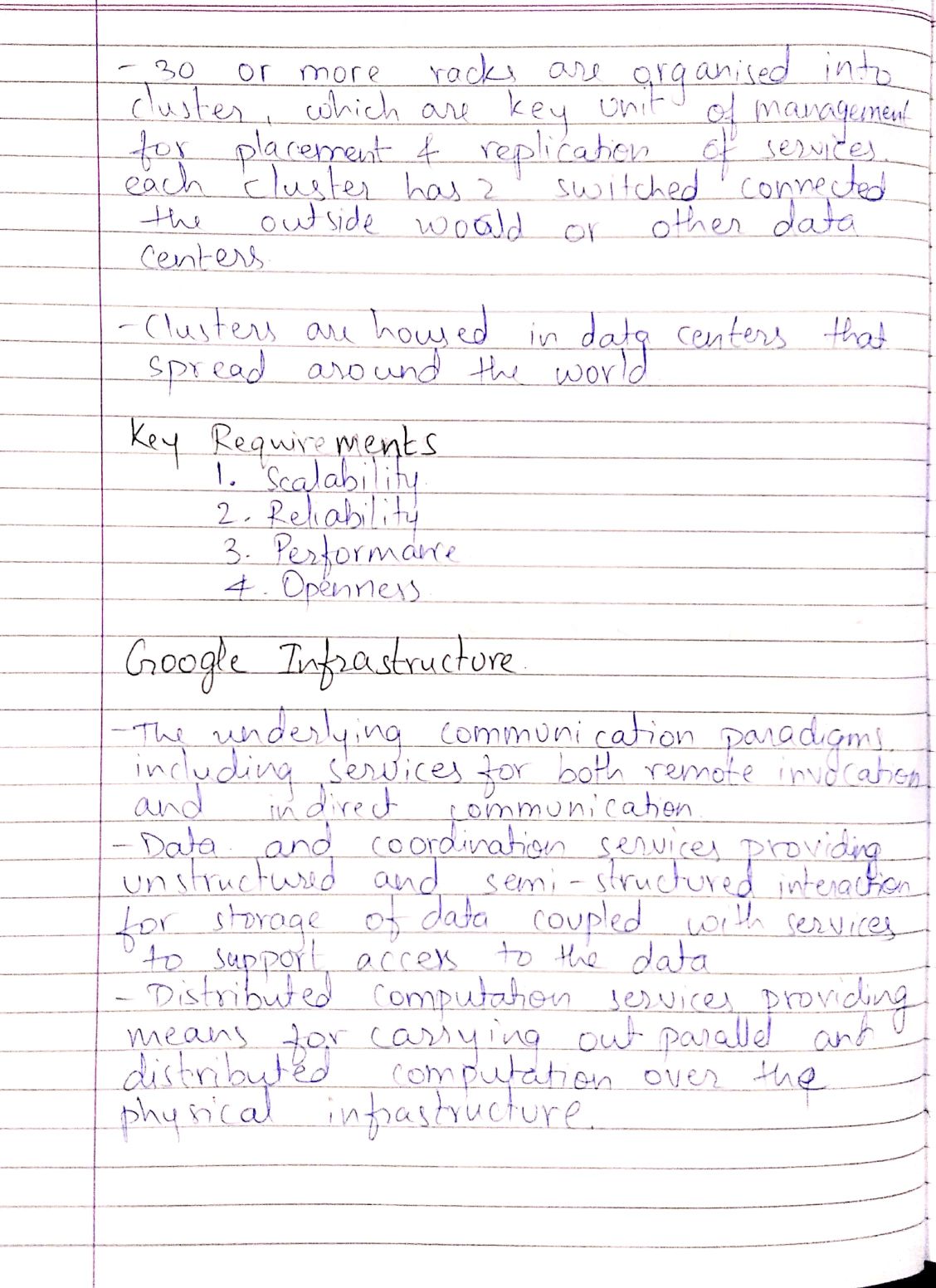
**Output :**



horizontal line

**Practical 12 : Case Study Google as a Distributed system.**

****

****