Vidya Vikas Education Trust's

Universal College of Engineering, Kaman Road, Vasai-401212

Accredited by B+ Grade by NAAC

Vision:

To be recognized globally as a department provides quality technical education that eventually caters to helping and serving the community

Mission:

To develop human resources with sound knowledge in theory and practice of computer science and engineering to motivate the students to solve real-world problems to help the society grow To provide a learning ambience to enhance innovations, team spirit and leadership qualities for students

LAB MANUAL

Mobile Application Development Lab

Name: Your Name

Class: BE Computer Engineering

Sem: VII

Roll No:

Your Roll

No

CSL702

Mobile Application Development Lab

Lab Outcome:

After successful completion of this course student will be able to:
1.Acquire practical knowledge within the chosen area of technology for project development.
2.Identify, discuss and justify the technical aspects of the chosen project with a comprehensive and systematic approach.

Description:

Design and implementation of any case study/ applications /experiments / mini project based on departmental level courses using modern tools.

Term work:

The distribution of marks for term work shall be as follows:
Lab/ Experimental Work: 15
Report/ Documentation: 05
Attendance (Theory & Practical): 05

Practical & Oral:

Examination is to be conducted based on respective departmental level courses by pair of internal and external examiners appointed by the University of Mumbai.

Lab Outcome

1.To develop and demonstrate mobile applications using various tools
2.Students will articulate the knowledge of GSM, CDMA & Bluetooth technologies and demonstrate it.
3.Students will able to carry out simulation of frequency re-use, hidden terminal problem
4.To develop security algorithms for mobile communication network
5.To demonstrate simulation and compare the performance of Wireless LAN
6.To implement and demonstrate mobile node discovery and route maintains.
Description: The software like Android Studio, J2ME, NS2, NS3 and any other software which is suitable are recommended for performing the practical.

List of Experiments

Sr. No.	Title	Page No.
1	To understand the cellular frequency reuse concept to find the co-channel cells for a particular cell. Design a game-based application on the above concept	4
2	Implementation a Bluetooth network with application as transfer of a file from one device to another.	7
3	To implement a basic function of Code Division Multiple Access (CDMA) to test the orthogonality and autocorrelation of a code to be used for CDMA operation. Write an application based on the above concept.	28
4	To implement Mobile node discovery.	33
5	Implementation of GSM security algorithms (A3/A5/A8)	42
6	Illustration of Hidden Terminal Problem (NS-2)	50
7	Develop an application that uses GUI components.	54
8	Write an application that draws basic graphical primitives on the screen.	61
9	Develop an application that makes use of database.	66
10	Implement an application that creates an alert upon receiving a message.	74

Experiment No.: 1

Aim: To understand the cellular frequency reuse concept to find the co-channel cells for a particular cell.

Theory:

In mobile communication systems a slot of a carrier frequency / code in a carrier frequency is a radio resource unit. This radio resource unit is assigned to a user in order to support a call/ session. The number of available such radio resources at a base station thus determines the number of users who can be supported in the call. Since in wireless channels a signal is "broadcast" i.e. received by all entities therefore one a resource is allocated to a user it cannot be reassigned until the user finished the call/ session. Thus, the number of users who can be supported in a wireless system is highly limited.

In order to support a large no. of users within a limited spectrum in a region the concept of frequency re-use is used.

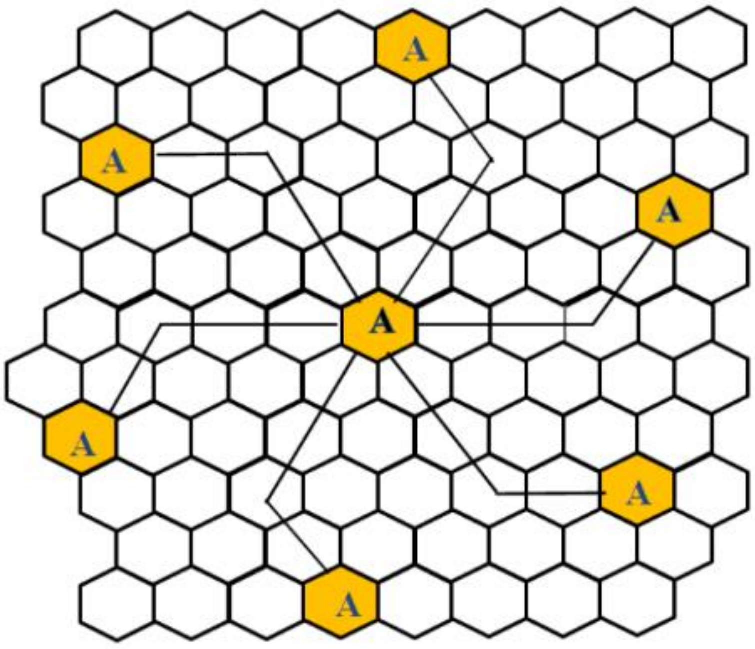
The signal radiated from the transmitter antenna gets attenuated with increasing distance. At a certain distance the signal strength falls below noise threshold and is no longer identifiable.

Cellular Frequency Reuse:

Each cellular base station is allocated a group of radio channels to be used within a small geographic area called a cell. Base stations in adjacent cells are assigned channel groups which contain completely different channels than neighboring cells. Base station antennas are designed to achieve the desired coverage within a particular cell. By limiting the coverage area within the boundaries of a cell, the same group of channels may be used to cover different cells that are separated from one another by geographic distances large enough to keep interference levels within tolerable limits. The design process of selecting and allocating channel groups for all cellular base stations within a system is called frequency reuse or frequency planning.

Co-channel Cells:

A larger cluster size causes the ratio between the cell radius and the distance between co-channel cells to decrease reducing co-channel interference. The value of N is a function of how much interference a mobile or base station can tolerate while maintaining a sufficient quality of communications. Since each hexagonal cell has six equidistant neighbors and the line joining the centers of any cell and each of its neighbors are separated by multiples of 60 degrees, only certain cluster sizes and cell layouts are possible. To connect without gaps between adjacent cells, the geometry of hexagons is such that the numbers of cells per cluster, N, can only have values that satisfy, `N=i^2+ij+j^2`



Method of locating co-channel cells in a cellular system. In this figure, N=19(i.e, i =3, j=2).

In this example N = 19 (i.e., i = 3, j = 2).

Where,

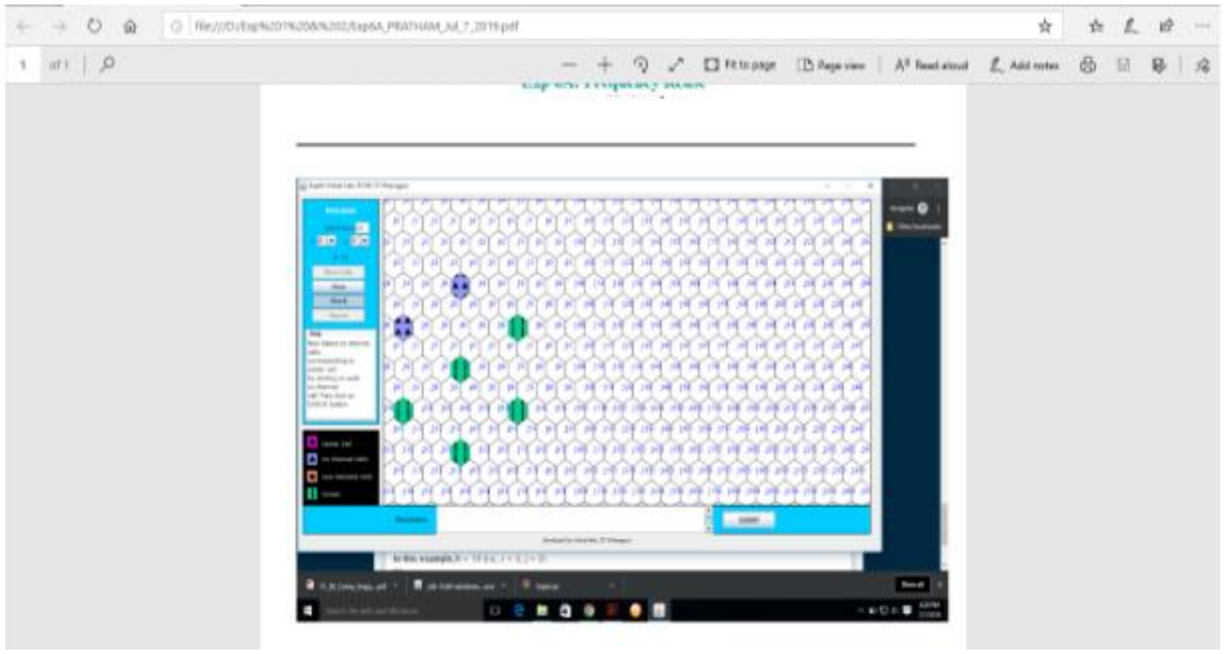
i and j are non-negative integers.

To nd the nearest co-channel neighbours of a particular cell,

a. move i cells along any chain of hexagons then,

b. turn 60 degrees counter-clockwise and move j cells.

Output:



Conclusion:

Thus, we have performed the frequency reuse experiment wherein we select a hexagon with particular frequency and then select other hexagons where the after which frequency can be used using above formula, we performed the experiment properly.

Experiment No.: 2

Aim: To implement a Bluetooth network with application as transfer of a file from one device to another.

Theory:

Bluetooth is a wireless technology standard used for exchanging data between fixed and mobile devices over short distances using UHF radio waves in the industrial, scientific and medical radio bands, from 2.402 GHz to 2.480 GHz, and building personal area networks (PANs). It was originally conceived as a wireless alternative to RS-232 data cables.
Bluetooth is managed by the Bluetooth Special Interest Group (SIG), which has more than 35,000 member companies in the areas of telecommunication, computing, networking, and consumer electronics. The IEEE standardized Bluetooth as IEEE 802.15.1, but no longer maintains the standard. The Bluetooth SIG oversees development of the specification, manages the qualification program, and protects the trademarks. A manufacturer must meet Bluetooth SIG standards to market it as a Bluetooth device.

Transfer of words between two phones using Bluetooth is done below.

Code:
Main_Activity.java:

```java
package com.example.bluetooth_communication;

import android.Manifest;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.Build;
import android.os.Bundle;

import android.util.Log;
import android.view.View;
import android.widget.AdapterView;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ListView;

import androidx.appcompat.app.AppCompatActivity;

import java.nio.charset.Charset;
import java.util.ArrayList;
import java.util.UUID;


public class MainActivity extends AppCompatActivity implements
AdapterView.OnItemClickListener{
   private static final String TAG = "MainActivity";

   BluetoothAdapter mBluetoothAdapter;
   Button btnEnableDisable_Discoverable;

   BluetoothConnectionService mBluetoothConnection;

   Button btnStartConnection;
   Button btnSend;

   EditText etSend;

   private static final UUID MY_UUID_INSECURE =
         UUID.fromString("8ce255c0-200a-11e0-ac64-0800200c9a66");

   BluetoothDevice mBTDevice;

   public ArrayList<BluetoothDevice> mBTDevices = new ArrayList<>();

   public DeviceListAdapter mDeviceListAdapter;

   ListView lvNewDevices;


   // Create a BroadcastReceiver for ACTION_FOUND
   private final BroadcastReceiver mBroadcastReceiver1 = new BroadcastReceiver() {
      public void onReceive(Context context, Intent intent) {
         String action = intent.getAction();
         // When discovery finds a device
         if (action.equals(mBluetoothAdapter.ACTION_STATE_CHANGED)) {
            final int state = intent.getIntExtra(BluetoothAdapter.EXTRA_STATE,
mBluetoothAdapter.ERROR);
```

```
                switch(state){
                    case BluetoothAdapter.STATE_OFF:
                        Log.d(TAG, "onReceive: STATE OFF");
                        break;
                    case BluetoothAdapter.STATE_TURNING_OFF:
                        Log.d(TAG, "mBroadcastReceiver1: STATE TURNING OFF");
                        break;
                    case BluetoothAdapter.STATE_ON:
                        Log.d(TAG, "mBroadcastReceiver1: STATE ON");
                        break;
                    case BluetoothAdapter.STATE_TURNING_ON:
                        Log.d(TAG, "mBroadcastReceiver1: STATE TURNING ON");
                        break;
                }
            }
        }
    };

    /**
     * Broadcast Receiver for changes made to bluetooth states such as:
     * 1) Discoverability mode on/off or expire.
     */
    private final BroadcastReceiver mBroadcastReceiver2 = new BroadcastReceiver() {

        @Override
        public void onReceive(Context context, Intent intent) {
            final String action = intent.getAction();

            if (action.equals(BluetoothAdapter.ACTION_SCAN_MODE_CHANGED)) {

                int mode = intent.getIntExtra(BluetoothAdapter.EXTRA_SCAN_MODE,
BluetoothAdapter.ERROR);

                switch (mode) {
                    //Device is in Discoverable Mode
                    case BluetoothAdapter.SCAN_MODE_CONNECTABLE_DISCOVERABLE:
                        Log.d(TAG, "mBroadcastReceiver2: Discoverability Enabled.");
                        break;
                    //Device not in discoverable mode
                    case BluetoothAdapter.SCAN_MODE_CONNECTABLE:
                        Log.d(TAG, "mBroadcastReceiver2: Discoverability Disabled. Able to
receive connections.");
                        break;
                    case BluetoothAdapter.SCAN_MODE_NONE:
                        Log.d(TAG, "mBroadcastReceiver2: Discoverability Disabled. Not able to
receive connections.");
                        break;
                    case BluetoothAdapter.STATE_CONNECTING:
                        Log.d(TAG, "mBroadcastReceiver2: Connecting....");
                        break;
                    case BluetoothAdapter.STATE_CONNECTED:
                        Log.d(TAG, "mBroadcastReceiver2: Connected.");
                        break;
                }

            }
        }
    };




    /**
     * Broadcast Receiver for listing devices that are not yet paired
     * -Executed by btnDiscover() method.
     */
    private BroadcastReceiver mBroadcastReceiver3 = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            final String action = intent.getAction();
            Log.d(TAG, "onReceive: ACTION FOUND.");

            if (action.equals(BluetoothDevice.ACTION_FOUND)){
                BluetoothDevice device = intent.getParcelableExtra
(BluetoothDevice.EXTRA_DEVICE);
                mBTDevices.add(device);
                Log.d(TAG, "onReceive: " + device.getName() + ": " + device.getAddress());
                mDeviceListAdapter = new DeviceListAdapter(context,
R.layout.device_adapter_view, mBTDevices);
                lvNewDevices.setAdapter(mDeviceListAdapter);
            }
        }
    };

    /**
     * Broadcast Receiver that detects bond state changes (Pairing status changes)
     */
    private final BroadcastReceiver mBroadcastReceiver4 = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            final String action = intent.getAction();

            if(action.equals(BluetoothDevice.ACTION_BOND_STATE_CHANGED)){
                BluetoothDevice mDevice =
intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
                //3 cases:
                //case1: bonded already
                if (mDevice.getBondState() == BluetoothDevice.BOND_BONDED){
                    Log.d(TAG, "BroadcastReceiver: BOND_BONDED.");
                    //inside BroadcastReceiver4
                    mBTDevice = mDevice;
                }
                //case2: creating a bone
                if (mDevice.getBondState() == BluetoothDevice.BOND_BONDING) {
                    Log.d(TAG, "BroadcastReceiver: BOND_BONDING.");
                }
                //case3: breaking a bond
                if (mDevice.getBondState() == BluetoothDevice.BOND_NONE) {
                    Log.d(TAG, "BroadcastReceiver: BOND_NONE.");
                }
            }
        }
    };



    @Override
    protected void onDestroy() {
        Log.d(TAG, "onDestroy: called.");
        super.onDestroy();
        unregisterReceiver(mBroadcastReceiver1);
        unregisterReceiver(mBroadcastReceiver2);
        unregisterReceiver(mBroadcastReceiver3);
        unregisterReceiver(mBroadcastReceiver4);
        //mBluetoothAdapter.cancelDiscovery();
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button btnONOFF = (Button)findViewById(R.id.btnONOFF);
        btnEnableDisable_Discoverable = (Button)findViewById(R.id.btnDiscoverable_on_off);
        lvNewDevices = (ListView)findViewById(R.id.lvNewDevices);
        mBTDevices = new ArrayList<>();

        btnStartConnection = (Button) findViewById(R.id.btnStartConnection);
        btnSend = (Button) findViewById(R.id.btnSend);
        etSend = (EditText) findViewById(R.id.editText);

        //Broadcasts when bond state changes (ie:pairing)
        IntentFilter filter = new
IntentFilter(BluetoothDevice.ACTION_BOND_STATE_CHANGED);
        registerReceiver(mBroadcastReceiver4, filter);

        mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();

        lvNewDevices.setOnItemClickListener(MainActivity.this);


        btnONOFF.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Log.d(TAG, "onClick: enabling/disabling bluetooth.");
                enableDisableBT();
            }
        });

        btnStartConnection.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                startConnection();
            }
        });

        btnSend.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                byte[] bytes = etSend.getText().toString().getBytes(Charset.defaultCharset());
                mBluetoothConnection.write(bytes);
            }
        });

    }

    //create method for starting connection
```

```
//***remember the conncction will fail and app will crash if you haven't paired first
    public void startConnection(){
        startBTConnection(mBTDevice,MY_UUID_INSECURE);
    }

    /**
     * starting chat service method
     */
    public void startBTConnection(BluetoothDevice device, UUID uuid){
        Log.d(TAG, "startBTConnection: Initializing RFCOM Bluetooth Connection.");

        mBluetoothConnection.startClient(device,uuid);
    }



    public void enableDisableBT(){
        if(mBluetoothAdapter == null){
            Log.d(TAG, "enableDisableBT: Does not have BT capabilities.");
        }
        if(!mBluetoothAdapter.isEnabled()){
            Log.d(TAG, "enableDisableBT: enabling BT.");
            Intent enableBTIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
            startActivity(enableBTIntent);

            IntentFilter BTIntent = new
IntentFilter(BluetoothAdapter.ACTION_STATE_CHANGED);
            registerReceiver(mBroadcastReceiver1, BTIntent);
        }
        if(mBluetoothAdapter.isEnabled()){
            Log.d(TAG, "enableDisableBT: disabling BT.");
            mBluetoothAdapter.disable();

            IntentFilter BTIntent = new
IntentFilter(BluetoothAdapter.ACTION_STATE_CHANGED);
            registerReceiver(mBroadcastReceiver1, BTIntent);
        }

    }


    public void btnEnableDisable_Discoverable(View view) {
        Log.d(TAG, "btnEnableDisable_Discoverable: Making device discoverable for 300
seconds.");
```

```
        Intent discoverableIntent = new
Intent(BluetoothAdapter.ACTION_REQUEST_DISCOVERABLE);
        discoverableIntent.putExtra(BluetoothAdapter.EXTRA_DISCOVERABLE_DURATION,
300);
        startActivity(discoverableIntent);

        IntentFilter intentFilter = new
IntentFilter(mBluetoothAdapter.ACTION_SCAN_MODE_CHANGED);
        registerReceiver(mBroadcastReceiver2,intentFilter);

    }

    public void btnDiscover(View view) {
        Log.d(TAG, "btnDiscover: Looking for unpaired devices.");

        if(mBluetoothAdapter.isDiscovering()){
            mBluetoothAdapter.cancelDiscovery();
            Log.d(TAG, "btnDiscover: Canceling discovery.");

            //check BT permissions in manifest
            checkBTPermissions();

            mBluetoothAdapter.startDiscovery();
            IntentFilter discoverDevicesIntent = new
IntentFilter(BluetoothDevice.ACTION_FOUND);
            registerReceiver(mBroadcastReceiver3, discoverDevicesIntent);
        }
        if(!mBluetoothAdapter.isDiscovering()){

            //check BT permissions in manifest
            checkBTPermissions();

            mBluetoothAdapter.startDiscovery();
            IntentFilter discoverDevicesIntent = new
IntentFilter(BluetoothDevice.ACTION_FOUND);
            registerReceiver(mBroadcastReceiver3, discoverDevicesIntent);
        }
    }

    /**
     * This method is required for all devices running API23+
     * Android must programmatically check the permissions for bluetooth. Putting the proper
permissions
```

```
     * in the manifest is not enough.
     *
     * NOTE: This will only execute on versions > LOLLIPOP because it is not needed otherwise.
     */
    private void checkBTPermissions() {
        if(Build.VERSION.SDK_INT > Build.VERSION_CODES.LOLLIPOP){
            int permissionCheck =
this.checkSelfPermission("Manifest.permission.ACCESS_FINE_LOCATION");
            permissionCheck +=
this.checkSelfPermission("Manifest.permission.ACCESS_COARSE_LOCATION");
            if (permissionCheck != 0) {

                this.requestPermissions(new
String[]{Manifest.permission.ACCESS_FINE_LOCATION,
Manifest.permission.ACCESS_COARSE_LOCATION}, 1001); //Any number
            }
        }else{
            Log.d(TAG, "checkBTPermissions: No need to check permissions. SDK version <
LOLLIPOP.");
        }
    }

    @Override
    public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {
        //first cancel discovery because its very memory intensive.
        mBluetoothAdapter.cancelDiscovery();

        Log.d(TAG, "onItemClick: You Clicked on a device.");
        String deviceName = mBTDevices.get(i).getName();
        String deviceAddress = mBTDevices.get(i).getAddress();

        Log.d(TAG, "onItemClick: deviceName = " + deviceName);
        Log.d(TAG, "onItemClick: deviceAddress = " + deviceAddress);

        //create the bond.
        //NOTE: Requires API 17+? I think this is JellyBean
        if(Build.VERSION.SDK_INT > Build.VERSION_CODES.JELLY_BEAN_MR2){
            Log.d(TAG, "Trying to pair with " + deviceName);
            mBTDevices.get(i).createBond();

            mBTDevice = mBTDevices.get(i);
            mBluetoothConnection = new BluetoothConnectionService(MainActivity.this);
        }
    }
}
```

BluetoothConnectionService.java:

```
package com.example.bluetooth_communication;

import android.app.ProgressDialog;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothServerSocket;
import android.bluetooth.BluetoothSocket;
import android.content.Context;
import android.util.Log;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.nio.charset.Charset;
import java.util.UUID;

/**
 * Created by User on 12/21/2016.
 */

public class BluetoothConnectionService {
    private static final String TAG = "BluetoothConnectionServ";

    private static final String appName = "MYAPP";

    private static final UUID MY_UUID_INSECURE =
            UUID.fromString("8ce255c0-200a-11e0-ac64-0800200c9a66");

    private final BluetoothAdapter mBluetoothAdapter;
    Context mContext;

    private AcceptThread mInsecureAcceptThread;

    private ConnectThread mConnectThread;
    private BluetoothDevice mmDevice;
    private UUID deviceUUID;
```

```
ProgressDialog mProgressDialog;

private ConnectedThread mConnectedThread;

public BluetoothConnectionService(Context context) {
    mContext = context;
    mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
    start();
}


/**
 * This thread runs while listening for incoming connections. It behaves
 * like a server-side client. It runs until a connection is accepted
 * (or until cancelled).
 */
private class AcceptThread extends Thread {

    // The local server socket
    private final BluetoothServerSocket mmServerSocket;

    public AcceptThread(){
        BluetoothServerSocket tmp = null;

        // Create a new listening server socket
        try{
            tmp =
mBluetoothAdapter.listenUsingInsecureRfcommWithServiceRecord(appName,
MY_UUID_INSECURE);

            Log.d(TAG, "AcceptThread: Setting up Server using: " +
MY_UUID_INSECURE);
        }catch (IOException e){
            Log.e(TAG, "AcceptThread: IOException: " + e.getMessage() );
        }

        mmServerSocket = tmp;
    }

    public void run(){
        Log.d(TAG, "run: AcceptThread Running.");

        BluetoothSocket socket = null;
```

```
            try{
                // This is a blocking call and will only return on a
                // successful connection or an exception
                Log.d(TAG, "run: RFCOM server socket start.....");

                socket = mmServerSocket.accept();

                Log.d(TAG, "run: RFCOM server socket accepted connection.");

            }catch (IOException e){
                Log.e(TAG, "AcceptThread: IOException: " + e.getMessage() );
            }

            //talk about this is in the 3rd
            if(socket != null){
                connected(socket,mmDevice);
            }

            Log.i(TAG, "END mAcceptThread ");
        }

        public void cancel() {
            Log.d(TAG, "cancel: Canceling AcceptThread.");
            try {
                mmServerSocket.close();
            } catch (IOException e) {
                Log.e(TAG, "cancel: Close of AcceptThread ServerSocket failed. " +
e.getMessage() );
            }
        }

    }

    /**
     * This thread runs while attempting to make an outgoing connection
     * with a device. It runs straight through; the connection either
     * succeeds or fails.
     */
    private class ConnectThread extends Thread {
        private BluetoothSocket mmSocket;

        public ConnectThread(BluetoothDevice device, UUID uuid) {
            Log.d(TAG, "ConnectThread: started.");
            mmDevice = device;
            deviceUUID = uuid;
        }

        public void run(){
            BluetoothSocket tmp = null;
            Log.i(TAG, "RUN mConnectThread ");

            // Get a BluetoothSocket for a connection with the
            // given BluetoothDevice
            try {
                Log.d(TAG, "ConnectThread: Trying to create InsecureRfcommSocket using
UUID: "
                        +MY_UUID_INSECURE );
                tmp = mmDevice.createRfcommSocketToServiceRecord(deviceUUID);
            } catch (IOException e) {
                Log.e(TAG, "ConnectThread: Could not create InsecureRfcommSocket " +
e.getMessage());
            }

            mmSocket = tmp;

            // Always cancel discovery because it will slow down a connection
            mBluetoothAdapter.cancelDiscovery();

            // Make a connection to the BluetoothSocket

            try {
                // This is a blocking call and will only return on a
                // successful connection or an exception
                mmSocket.connect();

                Log.d(TAG, "run: ConnectThread connected.");
            } catch (IOException e) {
                // Close the socket
                try {
                    mmSocket.close();
                    Log.d(TAG, "run: Closed Socket.");
                } catch (IOException e1) {
                    Log.e(TAG, "mConnectThread: run: Unable to close connection in socket " +
e1.getMessage());
                }
                Log.d(TAG, "run: ConnectThread: Could not connect to UUID: " +
MY_UUID_INSECURE );
            }
            //will talk about this in the 3rd video
            connected(mmSocket,mmDevice);
        }
        public void cancel() {
            try {
                Log.d(TAG, "cancel: Closing Client Socket.");
                mmSocket.close();
            } catch (IOException e) {
                Log.e(TAG, "cancel: close() of mmSocket in Connectthread failed. " +
e.getMessage());
            }
        }
    }



    /**
     * Start the chat service. Specifically start AcceptThread to begin a
     * session in listening (server) mode. Called by the Activity onResume()
     */
    public synchronized void start() {
        Log.d(TAG, "start");

        // Cancel any thread attempting to make a connection
        if (mConnectThread != null) {
            mConnectThread.cancel();
            mConnectThread = null;
        }
        if (mInsecureAcceptThread == null) {
            mInsecureAcceptThread = new AcceptThread();
            mInsecureAcceptThread.start();
        }
    }

    /**
     AcceptThread starts and sits waiting for a connection.
     Then ConnectThread starts and attempts to make a connection with the other devices
AcceptThread.
     **/

    public void startClient(BluetoothDevice device,UUID uuid){
        Log.d(TAG, "startClient: Started.");
        //initprogress dialog
        mProgressDialog = ProgressDialog.show(mContext,"Connecting Bluetooth"
                ,"Please Wait...",true);

        mConnectThread = new ConnectThread(device, uuid);
        mConnectThread.start();
    }

    /**
     Finally the ConnectedThread which is responsible for maintaining the BTConnection,
Sending the data, and
     receiving incoming data through input/output streams respectively.
     **/
    private class ConnectedThread extends Thread {
        private final BluetoothSocket mmSocket;
        private final InputStream mmInStream;
        private final OutputStream mmOutStream;

        public ConnectedThread(BluetoothSocket socket) {
            Log.d(TAG, "ConnectedThread: Starting.");

            mmSocket = socket;
            InputStream tmpIn = null;
            OutputStream tmpOut = null;

            //dismiss the progressdialog when connection is established
            try{
                mProgressDialog.dismiss();
            }catch (NullPointerException e){
                e.printStackTrace();
            }


            try {
                tmpIn = mmSocket.getInputStream();
                tmpOut = mmSocket.getOutputStream();
            } catch (IOException e) {
                e.printStackTrace();
            }
            mmInStream = tmpIn;
            mmOutStream = tmpOut;
        }

        public void run(){
```

```
        byte[] buffer = new byte[1024];  // buffer store for the stream

        int bytes; // bytes returned from read()

        // Keep listening to the InputStream until an exception occurs
        while (true) {
            // Read from the InputStream
            try {
                bytes = mmInStream.read(buffer);
                String incomingMessage = new String(buffer, 0, bytes);
                Log.d(TAG, "InputStream: " + incomingMessage);
            } catch (IOException e) {
                Log.e(TAG, "write: Error reading Input Stream. " + e.getMessage() );
                break;
            }
        }
    }

    //Call this from the main activity to send data to the remote device
    public void write(byte[] bytes) {
        String text = new String(bytes, Charset.defaultCharset());
        Log.d(TAG, "write: Writing to outputstream: " + text);
        try {
            mmOutStream.write(bytes);
        } catch (IOException e) {
            Log.e(TAG, "write: Error writing to output stream. " + e.getMessage() );
        }
    }

    /* Call this from the main activity to shutdown the connection */
    public void cancel() {
        try {
            mmSocket.close();
        } catch (IOException e) { }
    }
}

private void connected(BluetoothSocket mmSocket, BluetoothDevice mmDevice) {
    Log.d(TAG, "connected: Starting.");

    // Start the thread to manage the connection and perform transmissions
    mConnectedThread = new ConnectedThread(mmSocket);
    mConnectedThread.start();
}
```

```
    /**
     * Write to the ConnectedThread in an unsynchronized manner
     *
     * @param out The bytes to write
     * @see ConnectedThread#write(byte[])
     */
    public void write(byte[] out) {
        // Create temporary object
        ConnectedThread r;

        // Synchronize a copy of the ConnectedThread
        Log.d(TAG, "write: Write Called.");
        //perform the write
        mConnectedThread.write(out);
    }

}
```

DeviceListAdapter.java:

```
package com.example.bluetooth_communication;

import android.bluetooth.BluetoothDevice;
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.TextView;

import java.util.ArrayList;


public class DeviceListAdapter extends ArrayAdapter<BluetoothDevice> {

    private LayoutInflater mLayoutInflater;
    private ArrayList<BluetoothDevice> mDevices;
    private int  mViewResourceId;

    public DeviceListAdapter(Context context, int tvResourceId, ArrayList<BluetoothDevice>
devices){
        super(context, tvResourceId,devices);
        this.mDevices = devices;
        mLayoutInflater = (LayoutInflater)
context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        mViewResourceId = tvResourceId;
    }

    public View getView(int position, View convertView, ViewGroup parent) {
        convertView = mLayoutInflater.inflate(mViewResourceId, null);

        BluetoothDevice device = mDevices.get(position);

        if (device != null) {
            TextView deviceName = (TextView) convertView.findViewById(R.id.tvDeviceName);
            TextView deviceAdress = (TextView)
convertView.findViewById(R.id.tvDeviceAddress);

            if (deviceName != null) {
                deviceName.setText(device.getName());
            }
            if (deviceAdress != null) {
                deviceAdress.setText(device.getAddress());
            }
        }

        return convertView;
    }

}
```

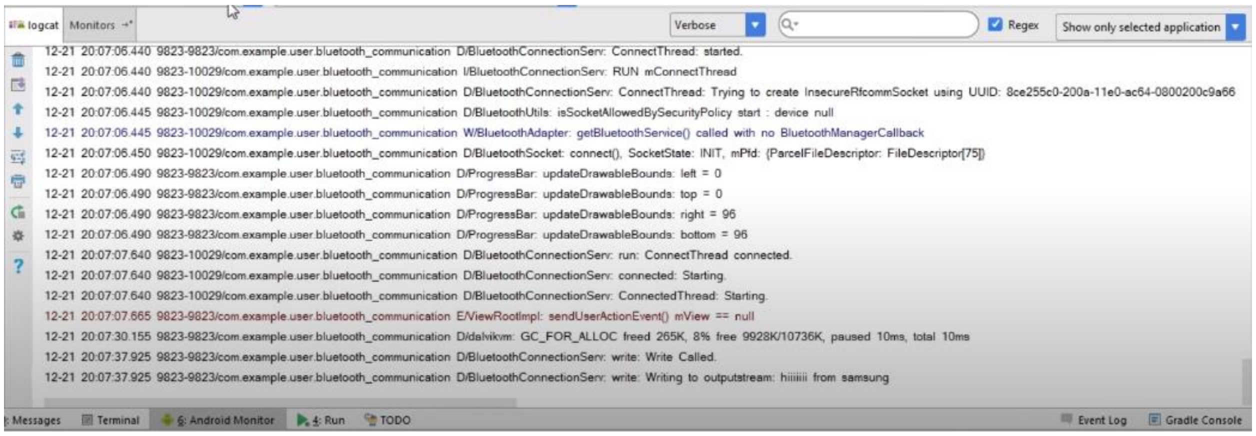
device_adapter_view.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/tvDeviceName"
        android:textSize="15sp"/>

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/tvDeviceAddress"
        android:textSize="15sp"/>

</LinearLayout>
```

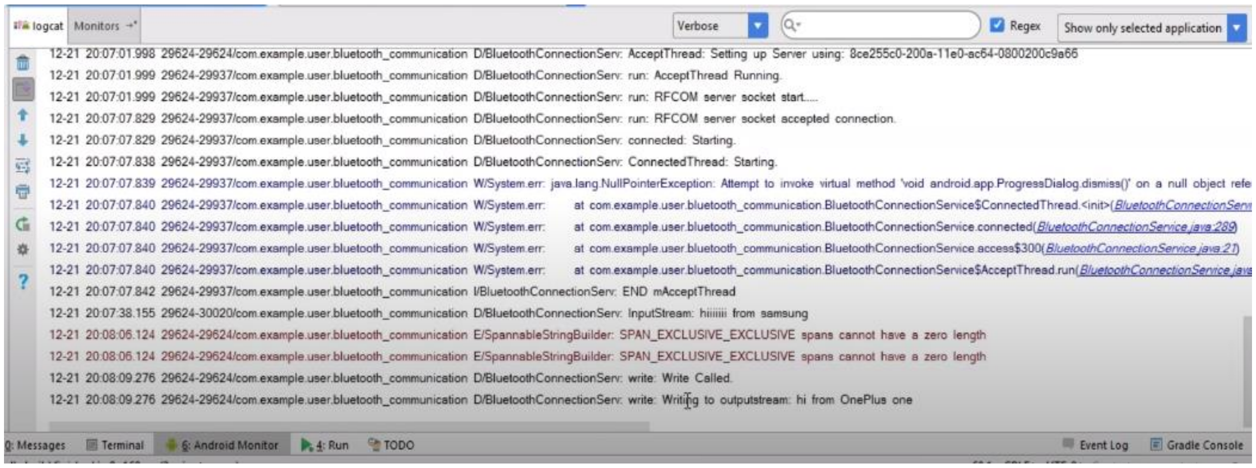
Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent">



    <Button
        android:id="@+id/btnONOFF"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:text="ON/OFF" />

    <Button
        android:text="Enable Discoverable"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/btnDiscoverable_on_off"
        android:onClick="btnEnableDisable_Discoverable"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"/>

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/btnFindUnpairedDevices"
        android:text="Discover"
        android:onClick="btnDiscover"/>

    <ListView
        android:layout_marginTop="15dp"
        android:layout_below="@+id/btnStartConnection"
        android:layout_width="match_parent"
        android:layout_height="150dp"
        android:id="@+id/lvNewDevices"/>

    <Button
        android:layout_marginTop="10dp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/btnFindUnpairedDevices"
        android:id="@+id/btnStartConnection"
        android:text="Start Connection"/>

    <EditText
        android:layout_width="250dp"
        android:layout_height="wrap_content"
        android:hint="Enter Text Here"
        android:layout_below="@+id/lvNewDevices"

        android:id="@+id/editText"/>

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="SEND"
        android:id="@+id/btnSend"
        android:layout_toRightOf="@+id/editText"
        android:layout_below="@+id/lvNewDevices"/>


</RelativeLayout>
```

Output:



```
12-21 20:07:06.440 9823-9823/com.example.user.bluetooth_communication D/BluetoothConnectionServ: ConnectThread: started.
12-21 20:07:06.440 9823-10029/com.example.user.bluetooth_communication I/BluetoothConnectionServ: RUN mConnectThread
12-21 20:07:06.440 9823-10029/com.example.user.bluetooth_communication D/BluetoothConnectionServ: ConnectThread: Trying to create InsecureRfcommSocket using UUID: 8ce255c0-200a-11e0-ac64-0800200c9a66
12-21 20:07:06.445 9823-10029/com.example.user.bluetooth_communication D/BluetoothUtils: isSocketAllowedBySecurityPolicy start : device null
12-21 20:07:06.445 9823-10029/com.example.user.bluetooth_communication W/BluetoothAdapter: getBluetoothService() called with no BluetoothManagerCallback
12-21 20:07:06.450 9823-10029/com.example.user.bluetooth_communication D/BluetoothSocket: connect(), SocketState: INIT, mPfd: {ParcelFileDescriptor: FileDescriptor[75]}
12-21 20:07:06.490 9823-9823/com.example.user.bluetooth_communication D/ProgressBar: updateDrawableBounds: left = 0
12-21 20:07:06.490 9823-9823/com.example.user.bluetooth_communication D/ProgressBar: updateDrawableBounds: top = 0
12-21 20:07:06.490 9823-9823/com.example.user.bluetooth_communication D/ProgressBar: updateDrawableBounds: right = 96
12-21 20:07:06.490 9823-9823/com.example.user.bluetooth_communication D/ProgressBar: updateDrawableBounds: bottom = 96
12-21 20:07:07.640 9823-10029/com.example.user.bluetooth_communication D/BluetoothConnectionServ: run: ConnectThread connected.
12-21 20:07:07.640 9823-10029/com.example.user.bluetooth_communication D/BluetoothConnectionServ: connected: Starting.
12-21 20:07:07.640 9823-10029/com.example.user.bluetooth_communication D/BluetoothConnectionServ: ConnectedThread: Starting.
12-21 20:07:07.665 9823-9823/com.example.user.bluetooth_communication E/ViewRootImpl: sendUserActionEvent() mView == null
12-21 20:07:30.155 9823-9823/com.example.user.bluetooth_communication D/dalvikvm: GC_FOR_ALLOC freed 265K, 8% free 9928K/10736K, paused 10ms, total 10ms
12-21 20:07:37.925 9823-9823/com.example.user.bluetooth_communication D/BluetoothConnectionServ: write: Write Called.
12-21 20:07:37.925 9823-9823/com.example.user.bluetooth_communication D/BluetoothConnectionServ: write: Writing to outputstream: hiiiiii from samsung
```



```
12-21 20:07:01.998 29624-29624/com.example.user.bluetooth_communication D/BluetoothConnectionServ: AcceptThread: Setting up Server using: 8ce255c0-200a-11e0-ac64-0800200c9a66
12-21 20:07:01.999 29624-29937/com.example.user.bluetooth_communication D/BluetoothConnectionServ: run: AcceptThread Running.
12-21 20:07:01.999 29624-29937/com.example.user.bluetooth_communication D/BluetoothConnectionServ: run: RFCOM server socket start.....
12-21 20:07:07.829 29624-29937/com.example.user.bluetooth_communication D/BluetoothConnectionServ: run: RFCOM server socket accepted connection.
12-21 20:07:07.829 29624-29937/com.example.user.bluetooth_communication D/BluetoothConnectionServ: connected: Starting.
12-21 20:07:07.838 29624-29937/com.example.user.bluetooth_communication D/BluetoothConnectionServ: ConnectedThread: Starting.
12-21 20:07:07.839 29624-29937/com.example.user.bluetooth_communication W/System.err: java.lang.NullPointerException: Attempt to invoke virtual method 'void android.app.ProgressDialog.dismiss()' on a null object refe
12-21 20:07:07.840 29624-29937/com.example.user.bluetooth_communication W/System.err:     at com.example.user.bluetooth_communication.BluetoothConnectionService$ConnectedThread.<init>(BluetoothConnectionServ
12-21 20:07:07.840 29624-29937/com.example.user.bluetooth_communication W/System.err:     at com.example.user.bluetooth_communication.BluetoothConnectionService.connected(BluetoothConnectionService.java:289)
12-21 20:07:07.840 29624-29937/com.example.user.bluetooth_communication W/System.err:     at com.example.user.bluetooth_communication.BluetoothConnectionService.access$300(BluetoothConnectionService.java:21)
12-21 20:07:07.840 29624-29937/com.example.user.bluetooth_communication W/System.err:     at com.example.user.bluetooth_communication.BluetoothConnectionService$AcceptThread.run(BluetoothConnectionService.java
12-21 20:07:07.842 29624-29937/com.example.user.bluetooth_communication I/BluetoothConnectionServ: END mAcceptThread
12-21 20:07:38.155 29624-30020/com.example.user.bluetooth_communication D/BluetoothConnectionServ: InputStream: hiiiiii from samsung
12-21 20:08:06.124 29624-29624/com.example.user.bluetooth_communication E/SpannableStringBuilder: SPAN_EXCLUSIVE_EXCLUSIVE spans cannot have a zero length
12-21 20:08:06.124 29624-29624/com.example.user.bluetooth_communication E/SpannableStringBuilder: SPAN_EXCLUSIVE_EXCLUSIVE spans cannot have a zero length
12-21 20:08:09.276 29624-29624/com.example.user.bluetooth_communication D/BluetoothConnectionServ: write: Write Called.
12-21 20:08:09.276 29624-29624/com.example.user.bluetooth_communication D/BluetoothConnectionServ: write: Writing to outputstream: hi from OnePlus one
```

Conclusion:

Thus, we have performed successfully the experiment of transferring data between two mobile phone using Bluetooth network and after that have checked and it performed.

Experiment No.: 3

Aim: To implement a basic function of Code Division Multiple Access (CDMA) to test the orthogonality and autocorrelation of a code to be used for CDMA operation. Write an application based on the above concept.

Theory:

Code-division multiple access (CDMA) is a channel access method used by various radio communication technologies. CDMA is an example of multiple access, where several transmitters can send information simultaneously over a single communication channel. This allows several users to share a band of frequencies (see bandwidth). To permit this without undue interference between the users, CDMA employs spread spectrum technology and a special coding scheme (where each transmitter is assigned a code).

CDMA is used as the access method in many mobile phone standards. IS-95, also called "cdmaOne", and its 3G evolution CDMA2000, are often simply referred to as "CDMA", but UMTS, the 3G standard used by GSM carriers, also uses "wideband CDMA", or W-CDMA, as well as TD-CDMA and TD-SCDMA, as its radio technologies.

The intended 4G successor to CDMA2000 was UMB (Ultra Mobile Broadband); however, in November 2008, Qualcomm announced it was ending development of the technology, favoring LTE instead

CDMA Orthogonality:

Techniques generally used are direct sequence spread spectrum modulation (DS-CDMA), frequency hopping or mixed CDMA detection (JDCDMA). Here, a signal is generated which extends over a wide bandwidth. A code called spreading code is used to perform this action. Using a group of codes, which are orthogonal to each other, it is possible to select a signal with a given code in the presence of many other signals with different orthogonal codes.

CDMA Autocorrelation:

Autocorrelation of the sequence, it determines the ability to synchronize and lock the spreading code for the received signal.

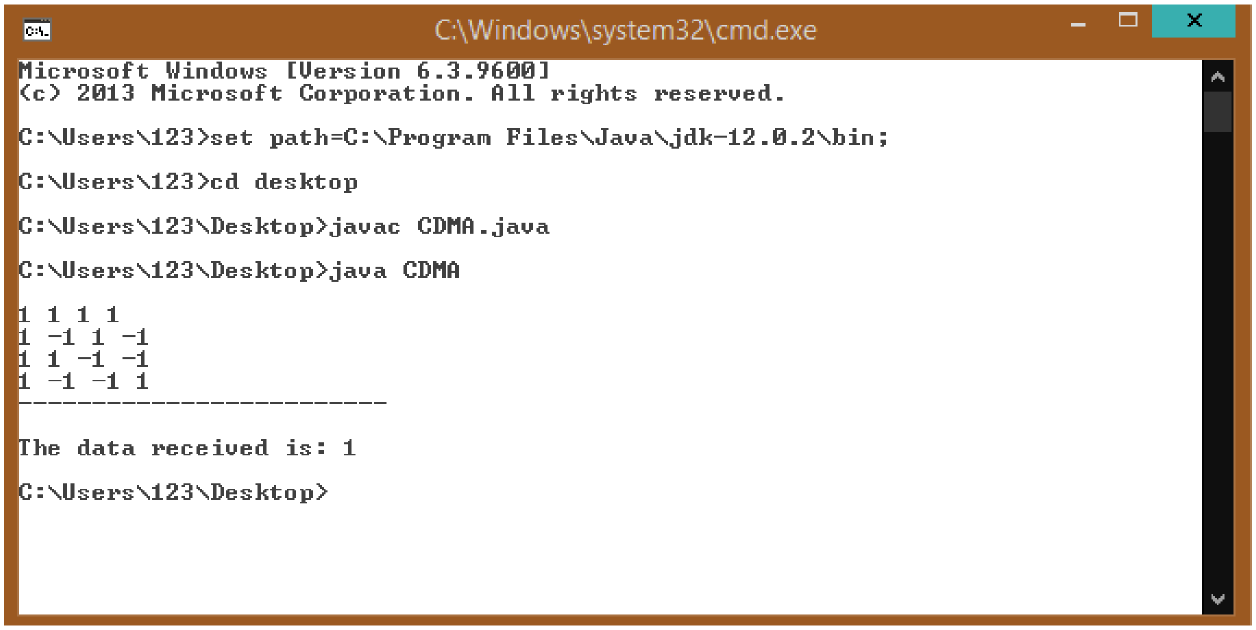
Code:

```
// Java code illustrating a simple implementation of CDMA

import java.util.*;

public class CDMA {
	private int[][] wtable;
	private int[][] copy;
	private int[] channel_sequence;

	public void setUp(int[] data, int num_stations)
	{

		wtable = new int[num_stations][num_stations];
		copy = new int[num_stations][num_stations];

		buildWalshTable(num_stations, 0, num_stations - 1, 0,
										num_stations - 1,
false);

		showWalshTable(num_stations);

		for (int i = 0; i < num_stations; i++) {

			for (int j = 0; j < num_stations; j++) {

				// Making a copy of walsh table
				// to be used later
				copy[i][j] = wtable[i][j];

				// each row in table is code for one station.
				// So we multiply each row with station data
				wtable[i][j] *= data[i];
			}
		}

		channel_sequence = new int[num_stations];

		for (int i = 0; i < num_stations; i++) {

			for (int j = 0; j < num_stations; j++) {
				// Adding all sequences to get channel sequence
				channel_sequence[i] += wtable[j][i];
			}
		}
	}
```

```
    public void listenTo(int sourceStation, int num_stations)
    {
        int innerProduct = 0;

        for (int i = 0; i < num_stations; i++) {

            // multiply channel sequence and source station code
            innerProduct += copy[sourceStation][i] * channel_sequence[i];
        }

        System.out.println("The data received is: " +
                                    (innerProduct / num_stations));
    }

    public int buildWalshTable(int len, int i1, int i2, int j1,
                                                        int j2, boolean
isBar)
    {
        // len = size of matrix. (i1, j1), (i2, j2) are
        // starting and ending indices of wtable.

        // isBar represents whether we want to add simple entry
        // or complement(southeast submatrix) to wtable.

        if (len == 2) {

            if (!isBar) {

                wtable[i1][j1] = 1;
                wtable[i1][j2] = 1;
                wtable[i2][j1] = 1;
                wtable[i2][j2] = -1;
            }
            else {

                wtable[i1][j1] = -1;
                wtable[i1][j2] = -1;
                wtable[i2][j1] = -1;
                wtable[i2][j2] = +1;
                }

                return 0;
            }

            int midi = (i1 + i2) / 2;
            int midj = (j1 + j2) / 2;

            buildWalshTable(len / 2, i1, midi, j1, midj, isBar);
            buildWalshTable(len / 2, i1, midi, midj + 1, j2, isBar);
            buildWalshTable(len / 2, midi + 1, i2, j1, midj, isBar);
            buildWalshTable(len / 2, midi + 1, i2, midj + 1, j2, !isBar);

            return 0;
        }

        public void showWalshTable(int num_stations)
        {

            System.out.print("\n");

            for (int i = 0; i < num_stations; i++) {
                for (int j = 0; j < num_stations; j++) {
                    System.out.print(wtable[i][j] + " ");
                }
                System.out.print("\n");
            }
            System.out.println("-------------------------");
            System.out.print("\n");
        }

        // Driver Code
        public static void main(String[] args)
        {
            int num_stations = 4;

            int[] data = new int[num_stations];

            //data bits corresponding to each station
            data[0] = -1;
```

```
                data[1] = -1;
                data[2] = 0;
                data[3] = 1;

                CDMA channel = new CDMA();

                channel.setUp(data, num_stations);

                // station you want to listen to
                int sourceStation = 3;

                channel.listenTo(sourceStation, num_stations);
        } }
```

Output:



```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\123>set path=C:\Program Files\Java\jdk-12.0.2\bin;

C:\Users\123>cd desktop

C:\Users\123\Desktop>javac CDMA.java

C:\Users\123\Desktop>java CDMA

1 1 1 1
1 -1 1 -1
1 1 -1 -1
1 -1 -1 1
-------------------------

The data received is: 1

C:\Users\123\Desktop>
```

Conclusion:

Thus, we have studied the CDMA code to test autocorrelation and orthogonality of codes and executed the same using the java code as above and got proper output for it.

Experiment No.: 4

Aim: To implement Mobile node discovery

Theory:

The mobile node is an end system or device such as a cell phone, PDA (Personal Digital assistant), or laptop whose software enables network roaming capabilities. A mobile node is an Internet-connected device whose location and point of attachment to the Internet may frequently be changed. This kind of node is often a cellular telephone or handheld or laptop computer, although a mobile node can also be a router. Special support is required to maintain Internet connections for a mobile node as it moves from one network or subnet to another, because traditional Internet routing assumes a device will always have the same IP address. Therefore, using standard routing procedures, a mobile user would have to change the device's IP address each time they connected through another network or subnet.
Since mobility and ease of connection are crucial considerations for mobile device users, organizations that want to promote mobile communications are putting a great deal of effort into making mobile connection and uncomplicated for the user. The Internet Engineering Task Force (IETF) Mobile IP working group has developed several standards or proposed standards to address these needs, including Mobile IP and later enhancements, Mobile IP version 6 (MIPv6) and Hierarchical Mobile IP version 6 (HMIPv6).

Code:

discover_device.java

```
package mypackage;

import java.io.IOException;
import javax.bluetooth.*;
import javax.bluetooth.DiscoveryListener;
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;

public class discover_device extends MIDlet implements CommandListener,DiscoveryListener {

private final List deviceList;
private final Command Exit,Refresh;
private String deviceName;
private DiscoveryAgent agent;
private Alert dialog;
```

```
public discover_device()
{
 deviceList = new List("List of Devices",List.IMPLICIT);
 Exit= new Command("Exit",Command.EXIT, 0);
 Refresh = new Command("Refresh",Command.SCREEN, 1);
  deviceList.addCommand(Exit);
 deviceList.addCommand(Refresh);
 deviceList.setCommandListener(this);
 Display.getDisplay(this).setCurrent(deviceList);
}

public void startApp() {
try {
deviceList.deleteAll();
LocalDevice local = LocalDevice.getLocalDevice();
local.setDiscoverable(DiscoveryAgent.GIAC);
deviceName = local.getFriendlyName();
agent = local.getDiscoveryAgent();
}
catch (BluetoothStateException ex) {
ex.printStackTrace();
}
try {
 agent.startInquiry(DiscoveryAgent.GIAC, this);
}
catch (BluetoothStateException ex) {
ex.printStackTrace();
}
}

public void pauseApp() {
}
public void destroyApp(boolean unconditional) {
}
public void commandAction(Command c, Displayable d) {
if(c==Exit)
{
this.destroyApp(true);
notifyDestroyed();
}
if(c==Refresh){
this.startApp();
}
}

public void deviceDiscovered(RemoteDevice btDevice, DeviceClass cod) {
String deviceaddress = null;
try {
deviceaddress = btDevice.getBluetoothAddress();//btDevice.getFriendlyName(true);
} catch (Exception ex) {
ex.printStackTrace();
}
deviceList.insert(0, deviceaddress , null);
}

public void servicesDiscovered(int transID, ServiceRecord[] servRecord) {
 throw new UnsupportedOperationException("Not supported yet.");
}

 public void serviceSearchCompleted(int transID, int respCode) {
        throw new UnsupportedOperationException("Not supported yet.");
    }




public void inquiryCompleted(int discType) {
        Alert dialog = null;
        if (discType != DiscoveryListener.INQUIRY_COMPLETED) {
            dialog = new Alert("Bluetooth Error","The inquiry failed to complete normally",null,
AlertType.ERROR);
         }
         else {
              dialog = new Alert("Inquiry Completed","The inquiry completed normally",
null,AlertType.INFO);
              }
              dialog.setTimeout(500);
              Display.getDisplay(this).setCurrent(dialog);
```

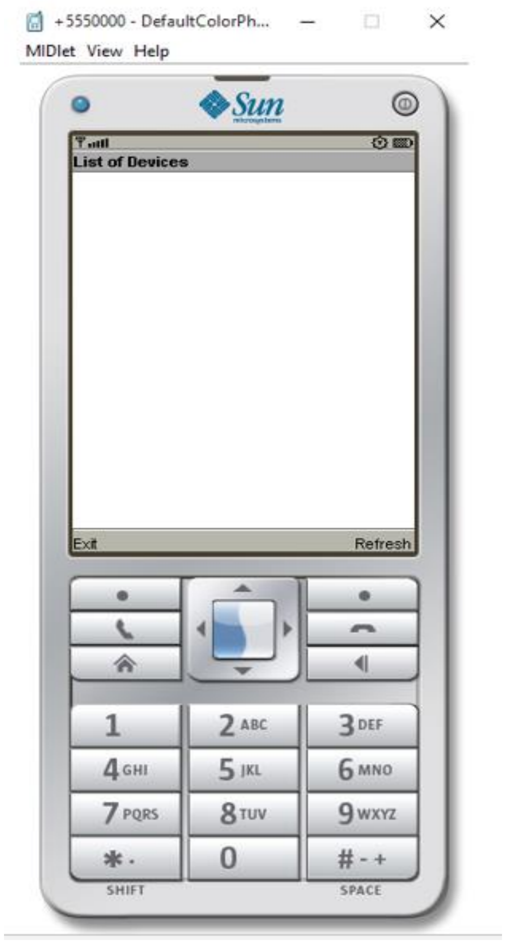
```
  }
}
```

Blue.java

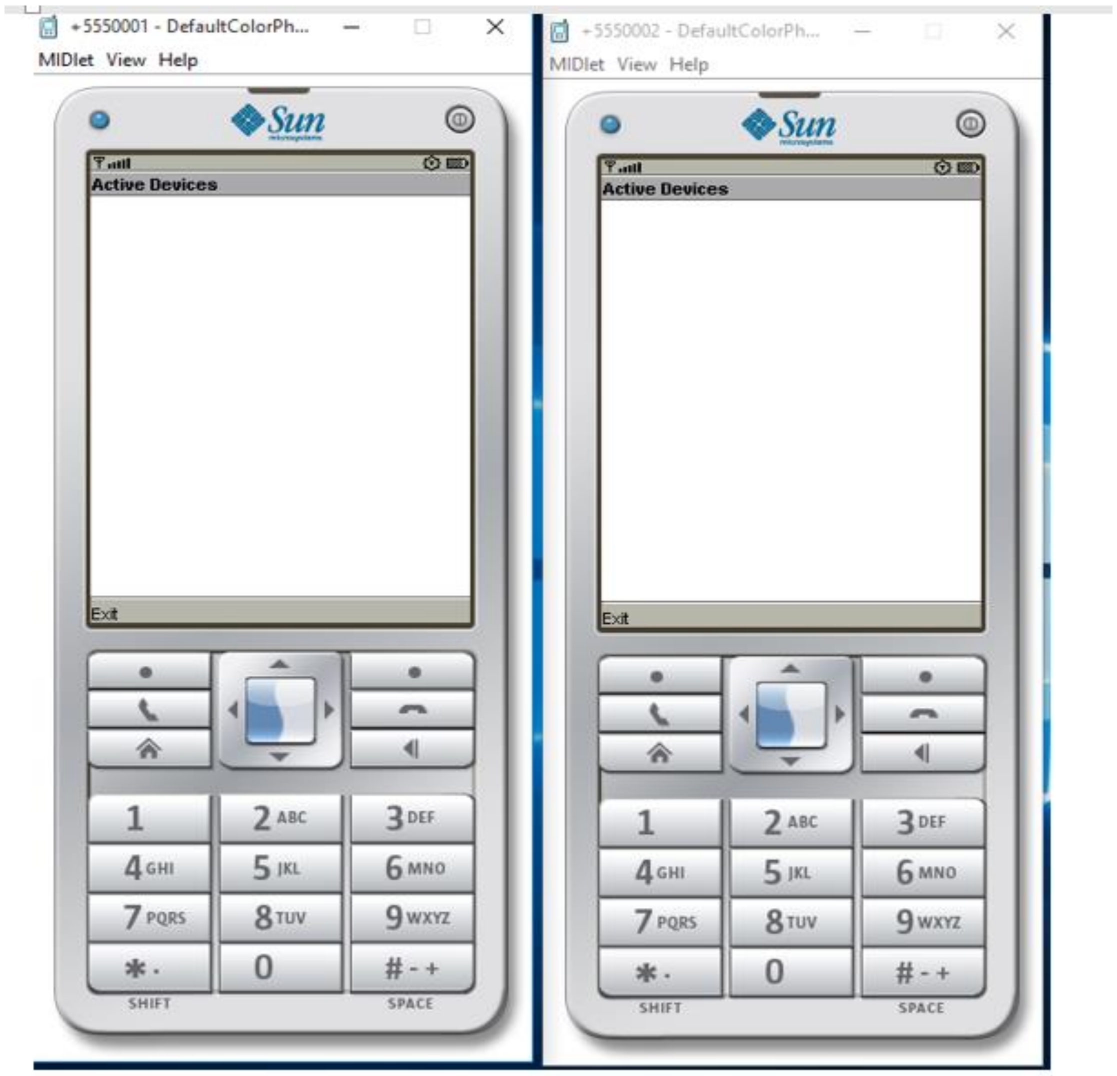
```
package mypackage;

import java.io.*;
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.io.*;
import javax.bluetooth.*;
import java.util.*;
public class Blue extends MIDlet implements CommandListener,DiscoveryListener
{
        private List activeDevices;
        private Command select,exit;
        private Display display;
        private LocalDevice local=null;
        private DiscoveryAgent agent = null;
        private Vector devicesFound = null;
        private ServiceRecord[] servicesFound = null;
        private String connectionURL = null;
        public void startApp() {
                display = Display.getDisplay(this);
                activeDevices = new List("Active Devices", List.IMPLICIT);
                select = new Command("Search Again", Command.OK, 0);
                exit = new Command("Exit", Command.EXIT, 0);
                activeDevices.addCommand(exit);
                activeDevices.setCommandListener(this);
                try {
                        local = LocalDevice.getLocalDevice(); }
                catch (Exception e) {}
                doDeviceDiscovery();
                display.setCurrent(activeDevices);
        }

        public void pauseApp() {}
        public void destroyApp(boolean unconditional) { notifyDestroyed(); }
        public void commandAction(Command cmd, Displayable disp) {
```

```
                        if (cmd == select && disp == activeDevices) {
                                activeDevices.deleteAll();
                                 doDeviceDiscovery();
                        }
                        if (cmd == exit) { destroyApp(false); }
        }
        public void inquiryCompleted(int param) {
                        try {
                        switch (param) {
                                case DiscoveryListener.INQUIRY_COMPLETED:

                                                if (devicesFound.size() > 0) {
                                                        activeDevices.addCommand(select);
                                                        activeDevices.setSelectCommand(select);
                                                }
                                                else { activeDevices.append("No Devices Found",
null); }
                                                break; }
                        }
                        catch (Exception e) {}
        }
        public void serviceSearchCompleted(int transID, int respCode) {}
        public void servicesDiscovered(int transID, ServiceRecord[] serviceRecord) {}
        public void deviceDiscovered(RemoteDevice remoteDevice, DeviceClass deviceClass)
{
                        String str = null;
                        try {
                                str = remoteDevice.getBluetoothAddress() + " - ";
                                str += remoteDevice.getFriendlyName(true);
                        } catch (Exception e) {}
                        activeDevices.append(str, null);
                        devicesFound.addElement(remoteDevice);
try {
        if (!agent.startInquiry(DiscoveryAgent.GIAC, this)) {}
} catch (BluetoothStateException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
}
         }
         private void doDeviceDiscovery() {
                try {
                            local = LocalDevice.getLocalDevice();
                            agent = local.getDiscoveryAgent();
                            devicesFound = new Vector();
                } catch (Exception e) {}
        }
}
```

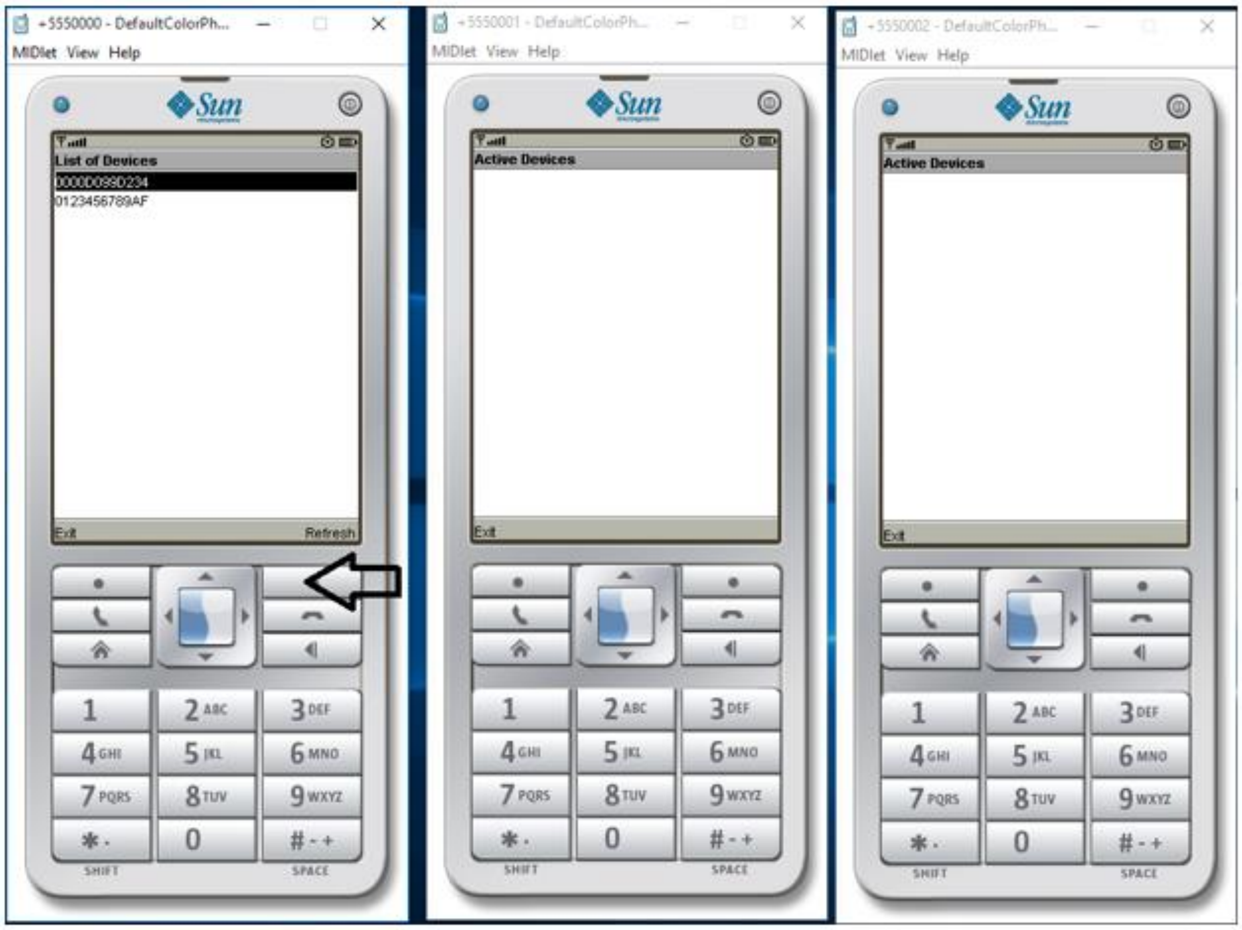
Output:
Step 1: First run discover_device.java. You will see this as the output. **(Hereafter referred as Device 0)**



Step 2: Run Blue.java file (You can run this file any number of times). For example, I have run this file for two times. So here I can see that two devices are running. **(Hereafter referred as Device 1 and Device 2 respectively).**



Step 3: Click in **Device 0** to the button pointed by the arrow. After clicking you will be able to see the Device number of **Device 1 & Device 2** as shown below.



```
0000D099D234
0123456789AF
```

Conclusion:

Thus, we have studied the mobile node discovery experiment properly and executed using the java code and after successfully running the code we have checked the output and it was prope

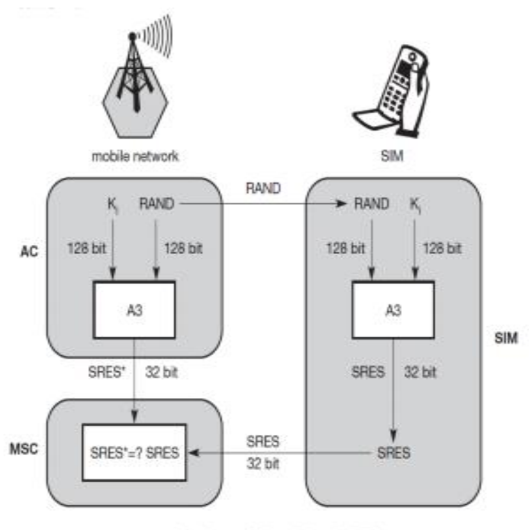
Experiment No.: 5

Aim: Implementation of GSM security algorithms (A3/A5/A8)

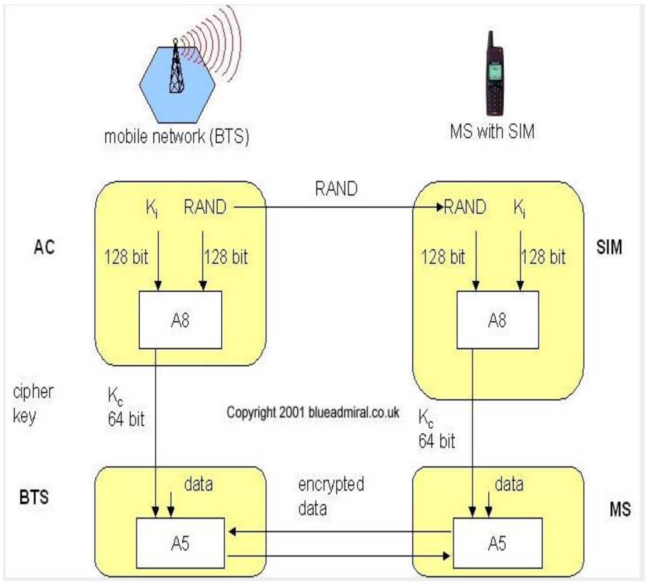
Theory:

The security procedures in GSM are aimed at protecting the network against unauthorized access and protecting the privacy of mobile subscriber against eavesdropping, eavesdropping on subscriber communication is prevented by ciphering the information. To protect identity and location of the subscriber the appropriate signaling channels are ciphered and Temporary Subscriber Identity (TMSI) instead of IMSI is used over the radio path. At the time of initiating a service, the mobile terminal is powered on the subscriber may be required to enter 4-8 digits Password Identification Number (PIN) to validate the ownership of the SIM. At the time of service provisioning the IMSI, the individual subscriber authentication key (Ki), the authentication algorithm (A3), the cipher key generation algorithm (A8) and the encryption algorithm (A5) are programmed into the SIM by GSM operator. The A3 ciphering algorithm is used to authenticate each mobile by verifying the user password within the SIM with the cryptographic key at the MSC. The A5 ciphering algorithm is used for encryption. It provides scrambling for 114 coded bits sent in each TS. The A8 is used for ciphering key. The IMSI and the secret authentication key (Ki) are specific to each mobile station, the authentication algorithm A3 and A8 are different for different networks and operator's encryption algorithm A5 is unique and needs to be used across all GSM network operators. The authentication center is responsible for all security aspects and its function is closely linked with HLR. The secret authentication key (Ki) is not known to mobile user and is the property of service provider, the home system of the mobile station (MS) generates the random number say Rand which is 126-bit number. This random number is sent to MS. The MS uses A3 algorithm to authenticate the user. The algorithm A3 uses Ki and Rand number to generate a signed

result called s_RES. MS sends s_RES to home system of MS. In the home system authentication contains Ki and it also uses the same authentication algorithm A3 to authenticate the valid user. The A3 algorithm use Ki and Rand generated by home system to generate a signed result called 〖(s〗 _RES). The s_RES generated by MS and authentication center are compared. If both s_RES are identical only then the user is valid and access is granted otherwise no



A3 Algorithm Block Diagram



Code:

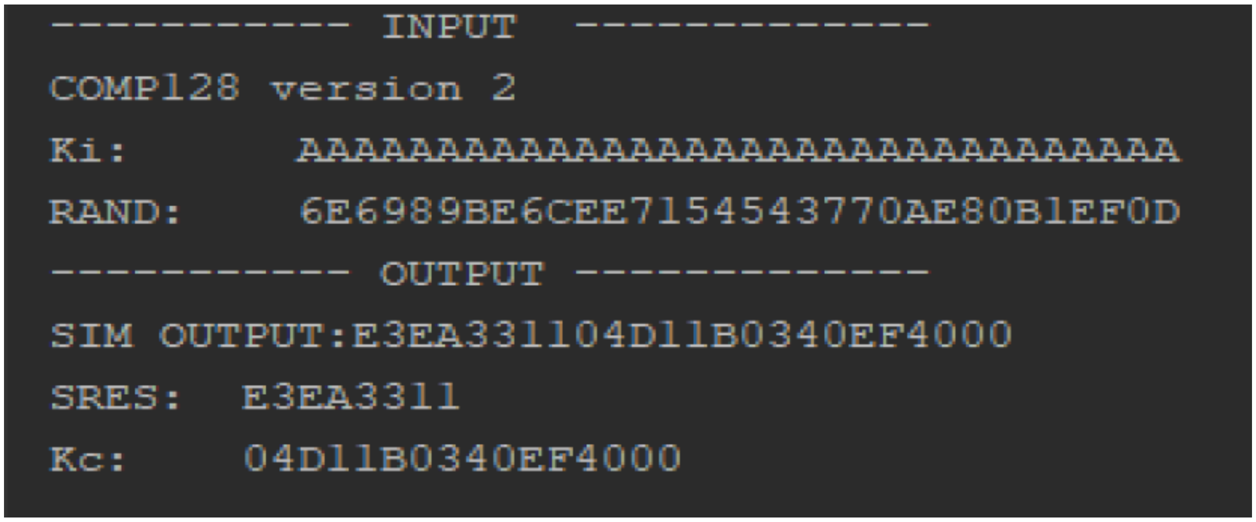
```
table0=[197, 235, 60, 151, 98, 96, 3, 100, 248, 118, 42, 117, 172, 211, 181, 203, 61,
        126, 156, 87, 149, 224, 55, 132, 186, 63, 238, 255, 85, 83, 152, 33, 160,
        184, 210, 219, 159, 11, 180, 194, 130, 212, 147, 5, 215, 92, 27, 46, 113,
        187, 52, 25, 185, 79, 221, 48, 70, 31, 101, 15, 195, 201, 50, 222, 137,
        233, 229, 106, 122, 183, 178, 177, 144, 207, 234, 182, 37, 254, 227, 231, 54,
        209, 133, 65, 202, 69, 237, 220, 189, 146, 120, 68, 21, 125, 38, 30, 2,
        155, 53, 196, 174, 176, 51, 246, 167, 76, 110, 20, 82, 121, 103, 112, 56,
        173, 49, 217, 252, 0, 114, 228, 123, 12, 93, 161, 253, 232, 240, 175, 67,
        128, 22, 158, 89, 18, 77, 109, 190, 17, 62, 4, 153, 163, 59, 145, 138,
        7, 74, 205, 10, 162, 80, 45, 104, 111, 150, 214, 154, 28, 191, 169, 213,
        88, 193, 198, 200, 245, 39, 164, 124, 84, 78, 1, 188, 170, 23, 86, 226,
        141, 32, 6, 131, 127, 199, 40, 135, 16, 57, 71, 91, 225, 168, 242, 206,
        97, 166, 44, 14, 90, 236, 239, 230, 244, 223, 108, 102, 119, 148, 251, 29,
        216, 8, 9, 249, 208, 24, 105, 94, 34, 64, 95, 115, 72, 134, 204, 43,
        247, 243, 218, 47, 58, 73, 107, 241, 179, 116, 66, 36, 143, 81, 250, 139,
        19, 13, 142, 140, 129, 192, 99, 171, 157, 136, 41, 75, 35, 165, 26 ]

table1=[170, 42, 95, 141, 109, 30, 71, 89, 26, 147, 231, 205, 239, 212, 124, 129, 216,
        79, 15, 185, 153, 14, 251, 162, 0, 241, 172, 197, 43, 10, 194, 235, 6,
        20, 72, 45, 143, 104, 161, 119, 41, 136, 38, 189, 135, 25, 93, 18, 224,
        171, 252, 195, 63, 19, 58, 165, 23, 55, 133, 254, 214, 144, 220, 178, 156,
        52, 110, 225, 97, 183, 140, 39, 53, 88, 219, 167, 16, 198, 62, 222, 76,
        139, 175, 94, 51, 134, 115, 22, 67, 1, 249, 217, 3, 5, 232, 138, 31,
        56, 116, 163, 70, 128, 234, 132, 229, 184, 244, 13, 34, 73, 233, 154, 179,
        131, 215, 236, 142, 223, 27, 57, 246, 108, 211, 8, 253, 85, 66, 245, 193,
        78, 190, 4, 17, 7, 150, 127, 152, 213, 37, 186, 2, 243, 46, 169, 68,
        101, 60, 174, 208, 158, 176, 69, 238, 191, 90, 83, 166, 125, 77, 59, 21,
        92, 49, 151, 168, 99, 9, 50, 146, 113, 117, 228, 65, 230, 40, 82, 54,
        237, 227, 102, 28, 36, 107, 24, 44, 126, 206, 201, 61, 114, 164, 207, 181,
        29, 91, 64, 221, 255, 48, 155, 192, 111, 180, 210, 182, 247, 203, 148, 209,
        98, 173, 11, 75, 123, 250, 118, 32, 47, 240, 202, 74, 177, 100, 80, 196,
        33, 248, 86, 157, 137, 120, 130, 84, 204, 122, 81, 242, 188, 200, 149, 226,
        218, 160, 187, 106, 35, 87, 105, 96, 145, 199, 159, 12, 121, 103, 112]


def comp128v23_internal(KXOR,RAND):
"""Internal part of the COMP128v23 algo, should not be called manually
"""
temp = [0] * 16
KM_RM = RAND + KXOR


for i in range(5):
for z in range(16):
temp[z] = table0[table1[KM_RM[16+z]] ^ KM_RM[z] ]


j = 0
while ( (1 << i) > j):
k = 0
while ( (1 << (4 - i)) > k ):
    KM_RM[((2 * k + 1) << i )+j] = table0[table1[temp[(k << i) + j]] ^
(KM_RM[(k << i) + 16 + j])]
    KM_RM[ (k << (i + 1)) + j] = temp[(k << i) + j]
    k = k+1
j = j + 1


output = [0]*16


for i in range(16):
for j in range(8):
output[i] = output[i] ^ (((KM_RM[(19 * (j + 8 * i) + 19) % 256 / 8] >> (3
* j + 3) % 8) & 1) << j)


return output
```

```
def comp128v23(K, RAND, version = 2):
    """The entry point for COMP128v2 and COMP128v3 algorithm
    K = The secret Ki number (that should be inside of your SIM card) - Format: list of integers
    RAND = The random number generated by the tower - Format: list of integers
    version = Version selecting integer (can be 2 or 3) - Format: integer
    """

    assert version in [2,3] , "This function only support COMP128 version 2 and 3!"
    assert len(K) == 16     , "Ki incorrect (length must be 16)"
    assert len(RAND) == 16  , "RAND incorrect (length must be 16)"

    K_MIX = [0]*16
    RAND_MIX = [0]*16
    KATYVASZ = [0]*16
    output = [0]*16

    for i in range(8):
        K_MIX[i] = K[15 - i]
        K_MIX[15 - i] = K[i]

    for i in range(8):
        RAND_MIX[i] = RAND[15 - i]
        RAND_MIX[15 - i] = RAND[i]

    for i in range(16):
        KATYVASZ[i] = K_MIX[i] ^ RAND_MIX[i]


for i in range(8):
        RAND_MIX = comp128v23_internal(KATYVASZ,RAND_MIX)


for i in range(16):
        output[i] = RAND_MIX[15-i]




if version == 2:
        output[15] = 0
        output[14] = 4 * (output[14] >> 2)


s = 8
i = 0
while i < 4:
        output[s+i-4] = output[s+i]
        output[s+i] = output[s+i+4]
        i = i+1


#the algorithm uses 16 bytes until this point, but only 12 bytes are effective
#also 12 bytes coming out from the SIM card


output_final = output[:12]
return output_final
```

```
def hex2intarr(input):
        """converts hex string to an array of integers
        """
        return map(lambda a: int(a.encode('hex'),16), (a for a in input.decode('hex')))


def intarr2hex(input):
        """converts array of integers to hex strings
        """
        return ''.join('{:02x}'.format(x) for x in input).upper()


if __name__ == '__main__':
        import argparse


        parser = argparse.ArgumentParser(description='Process some integers.')
        parser.add_argument('Ki', metavar='Ki', default = "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA" ,nargs='?', help='The super secret Ki key')
        parser.add_argument('RAND', metavar='RAND', default = "6E6989BE6CEE7154543770AE80B1EF0D", nargs='?', help='The RANDom number you recieve from the tower')
        parser.add_argument('version', metavar='version', default = 2, nargs='?', help='The version of the COMP128 algo you wish to use (options: 2 or 3)')


        args = parser.parse_args()


        Ki = hex2intarr(args.Ki)
        RAND = hex2intarr(args.RAND)
```

```
version = args.version

print '----------- INPUT  -------------'
print 'COMP128 version ' + str(version)
print 'Ki:      ' + intarr2hex(Ki)
print 'RAND:    ' + intarr2hex(RAND)

OUTPUT = comp128v23(Ki, RAND, version)
SRES = OUTPUT[:4]
Kc = OUTPUT[4:]

print '----------- OUTPUT -------------'
print "SIM OUTPUT:" + intarr2hex(OUTPUT)
print "SRES:  " + intarr2hex(SRES)
print "Kc:    " + intarr2hex(Kc)
```

Output:



```
----------- INPUT  -------------
COMP128 version 2
Ki:      AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
RAND:    6E6989BE6CEE7154543770AE80B1EF0D
----------- OUTPUT -------------
SIM OUTPUT:E3EA331104D11B0340EF4000
SRES:  E3EA3311
Kc:    04D11B0340EF4000
```

Conclusion: Thus, we have performed the experiment to implement the A3, A5 and A8 algorithm of the GSM security and we have implemented and executed it properly with proper output.

Experiment No.: 6

Aim: Illustration of Hidden Terminal Problem (NS-2)

Theory:

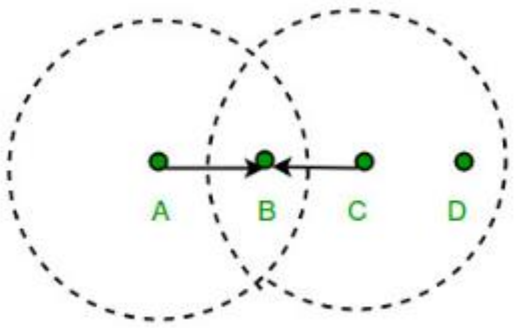
A wireless network with lack of centralized control entity, sharing of wireless bandwidth among network access nodes i.e. medium access control (MAC) nodes must be organized in decentralized manner. The hidden terminal problem occurs when a terminal is visible from a wireless access point (APs), but not from other nodes communicating with that AP. This situation leads the difficulties in medium access control sublayer over wireless networking.

In a formal way hidden terminal are nodes in a wireless network that are out of range of other node or a collection of nodes. Consider a wireless networking, each node at the far edge of the access point's range, which is known as A, can see the access point, but it is unlikely that the same node can see a node on the opposite end of the access point's range, C. These nodes are known as hidden. The problem is when nodes A and C start to send packets simultaneously to the access point B. Because the nodes A and C are out of range of each other and so cannot detect a collision while transmitting, Carrier sense multiple access with collision detection (CSMA/CD) does not work, and collisions occur, which then corrupt the data received by the access point. To overcome the hidden node problem, RTS/CTS handshaking (IEEE 802.11 RTS/CTS) is implemented in conjunction with the Carrier sense multiple accesses with collision avoidance (CSMA/CA) scheme. The same problem exists in a MANET.

The transmission range of access point A reaches at B, but not at access point C, similarly transmission range of access point C reaches B, but not at A. These nodes are known as hidden terminals. The problem occurs when nodes A and C start to send data packets simultaneously to the access point B. Because the access points A and C are out of range of each other and resultant they cannot detect a collision while transmitting, Carrier sense multiple access with collision detection (CSMA/CD) does not work, and collisions occur, which then corrupt the data received by the access point B due to the hidden terminal problem.

The hidden terminal analogy is described as follows:

- Terminal A sends data to B, terminal C cannot hear A
- Terminal C wants to send data to B, terminal C senses a "free" medium (CS fails) and starts transmitting
- Collision at B occurs, A cannot detect this collision (CD fails) and continues with its transmission to B
- Terminal A is "hidden" from C and vice versa.



Hidden Node Problem

The solution of hidden terminal problem is as follows.

When A wants to send a packet to B, A first sends a Request-to-send (RTS) to B.

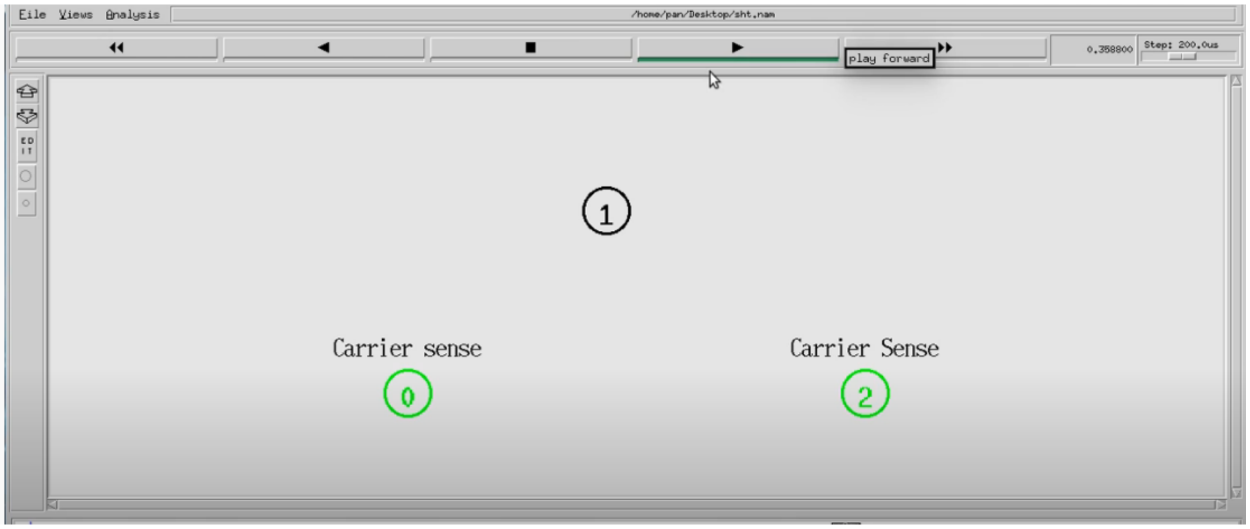
On receiving RTS, B responds by sending Clear-to-Send (CTS).

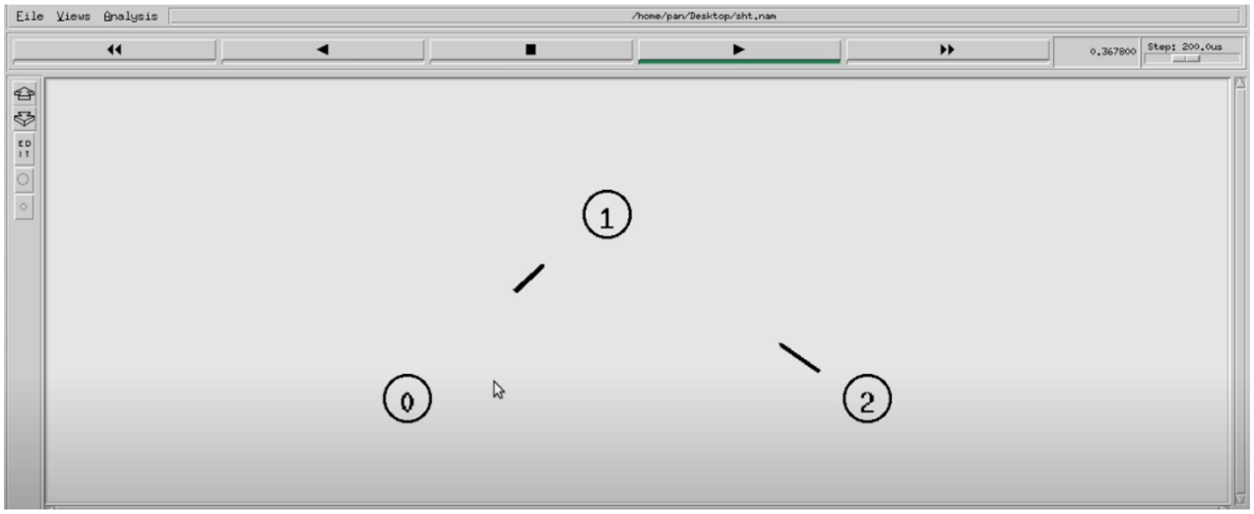
When C overhears a CTS, it keeps quiet for the duration of the transfer.

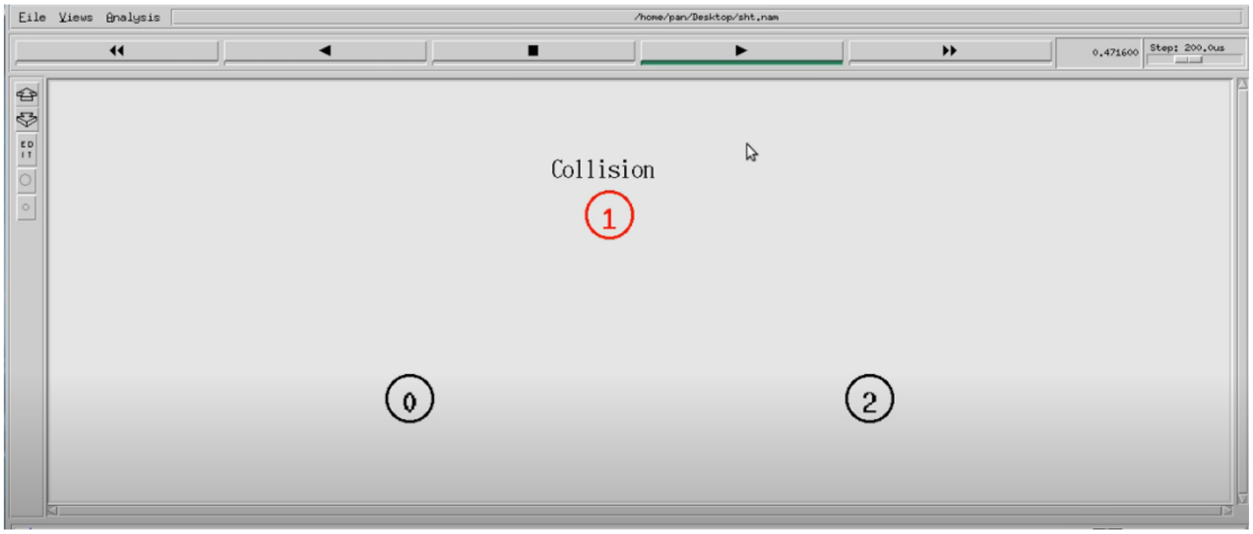
Transfer duration is included in both RTS and CTS.

RTS and CTS are short frames, reduces collision chance.

Output:
1) The node 0 and 2 want to send data to node 1 the range of node 0 and 2 is limited to 1 they do not know that other node is also sending data to 1 and therefore collision occurs.


File Views Analysis
/home/pan/Desktop/sht.nam
play forward
0.358800
Step: 200.0us
1
Carrier sense
0
Carrier Sense
2


File Views Analysis
/home/pan/Desktop/sht.nam
0.367800
Step: 200.0us
1
0
2


File Views Analysis
/home/pan/Desktop/sht.nam
0.471600
Step: 200.0us
Collision
1
0
2

Conclusion:

Thus, we have performed the experiment of and illustrated the hidden terminal problem using NS2 and properly explained the same which helps to understand better.

Experiment No. 7

Aim: Develop an application that uses GUI components.

Theory:

A typical user interface of an android application consists of action bar and the application content area.

- Main Action Bar
- View Control
- Content Area
- Split Action Bar

The basic unit of android application is the activity. A UI is defined in an xml file. During compilation, each element in the XML is compiled into equivalent Android GUI class with attributes represented by methods.

View and ViewGroups

An activity is consist of views. A view is just a widget that appears on the screen. It could be button etc. One or more views can be grouped together into one GroupView. Example of ViewGroup includes layouts.

Types of layout

There are many types of layout. Some of which are listed below −

- Linear Layout
- Absolute Layout
- Table Layout
- Frame Layout
- Relative Layout

The basic building block for user interface is a **View** object which is created from the View class and occupies a rectangular area on the screen and is responsible for drawing and event handling. View is the base class for widgets, which are used to create interactive UI components like buttons, text fields, etc.

The **ViewGroup** is a subclass of **View** and provides invisible container that hold other Views or other ViewGroups and define their layout properties.

At third level we have different layouts which are subclasses of ViewGroup class and a typical layout defines the visual structure for an Android user interface and can be created either at run time using **View/ViewGroup** objects or you can declare your layout using simple XML file **main_layout.xml** which is located in the res/layout folder of your project.

Code:

MainActivity.java

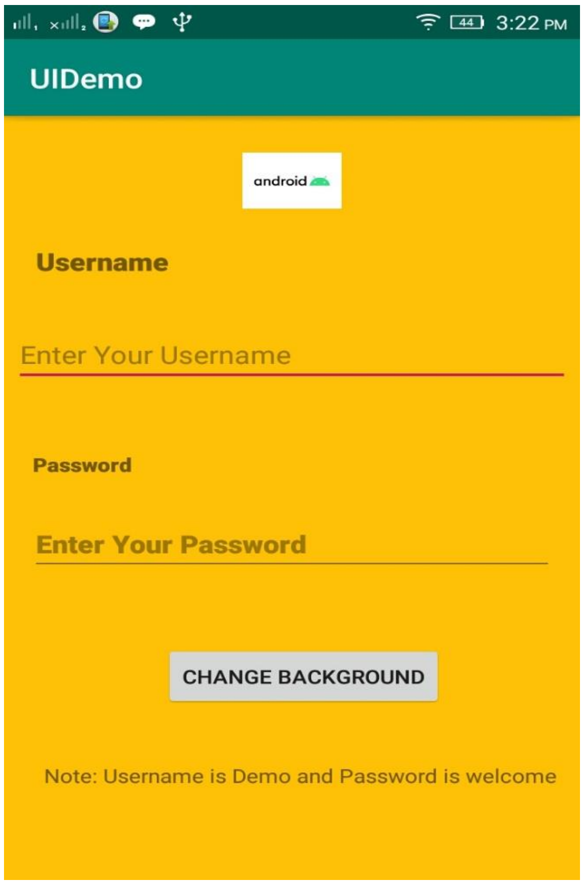
```java
package com.example.uidemo;

import androidx.appcompat.app.AppCompatActivity;
import androidx.constraintlayout.widget.ConstraintLayout;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.RelativeLayout;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    Button btnAuthenticate;
    EditText edtUsername, edtPassword;
    ConstraintLayout canvas;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        btnAuthenticate = (Button)findViewById(R.id.btnAuthenticate);
        edtUsername = (EditText)findViewById(R.id.edtUsername);
        edtPassword = (EditText)findViewById(R.id.edtPassword);
        canvas = (ConstraintLayout) findViewById(R.id.canvas);

        btnAuthenticate.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String username = null, password = null;
                username=edtUsername.getText().toString();
                password=edtPassword.getText().toString();
            if(username.equals("Demo") && password.equals("welcome")){
                Toast.makeText(getApplicationContext(),"User Authenticated",
Toast.LENGTH_LONG).show();
                canvas.setBackgroundResource(R.color.authenticatedColor);
            } else {
                Toast.makeText(getApplicationContext(),"Please enter correct username or
password",Toast.LENGTH_LONG).show();
            }
        }
    });
  }
}
```

Activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:id="@+id/canvas"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:background="#FFC107"
  tools:context=".MainActivity">

  TextView
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:text="Hello World!"
  app:layout_constraintBottom_toBottomOf="parent"
  app:layout_constraintLeft_toLeftOf="parent"
  app:layout_constraintRight_toRightOf="parent"
  app:layout_constraintTop_toTopOf="parent" />

  <ImageView
    android:id="@+id/imageView2"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_marginStart="24dp"
    android:layout_marginTop="27dp"
    android:layout_marginEnd="24dp"
    android:layout_marginBottom="26dp"
    app:layout_constraintBottom_toTopOf="@+id/textView2"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:srcCompat="@drawable/android" />

<TextView
    android:id="@+id/textView2"
    android:layout_width="128dp"
    android:layout_height="0dp"
    android:layout_marginStart="20dp"
    android:layout_marginBottom="21dp"
    android:fontFamily="sans-serif-black"
    android:text="Username"
    android:textSize="18sp"
    android:textStyle="bold"
    app:layout_constraintBottom_toTopOf="@+id/edtUsername"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/imageView2" />

<EditText
    android:id="@+id/edtUsername"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_marginStart="6dp"
    android:layout_marginEnd="6dp"
    android:layout_marginBottom="48dp"
    android:ems="10"
    android:hint="Enter Your Username"
    android:inputType="textPersonName"
    app:layout_constraintBottom_toTopOf="@+id/textView3"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView2" />

<TextView
    android:id="@+id/textView3"
    android:layout_width="153dp"
    android:layout_height="0dp"
    android:layout_marginStart="2dp"
    android:layout_marginBottom="8dp"
    android:fontFamily="sans-serif-black"
    android:text="Password"
    android:textSize="14sp"
    android:textStyle="bold"
    app:layout_constraintBottom_toTopOf="@+id/edtPassword"
    app:layout_constraintStart_toStartOf="@+id/edtPassword"
    app:layout_constraintTop_toBottomOf="@+id/edtUsername" />

<EditText
    android:id="@+id/edtPassword"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_marginStart="16dp"

    android:layout_marginEnd="16dp"
    android:layout_marginBottom="50dp"
    android:ems="10"
    android:fontFamily="sans-serif-black"
    android:hint="Enter Your Password"
    android:inputType="textPassword"
    app:layout_constraintBottom_toTopOf="@+id/btnAuthenticate"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView3" />

<Button
    android:id="@+id/btnAuthenticate"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="39dp"
    android:text="Change Background"
    app:layout_constraintBottom_toTopOf="@+id/textView"
    app:layout_constraintEnd_toEndOf="@+id/textView"
    app:layout_constraintStart_toStartOf="@+id/textView"
    app:layout_constraintTop_toBottomOf="@+id/edtPassword" />

<TextView
    android:id="@+id/textView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="25dp"
    android:layout_marginBottom="67dp"
    android:text="Note: Username is Demo and Password is welcome "
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/btnAuthenticate" />20dp"
```

```
   tools:layout_editor_absoluteY="644dp" />20dp"
   tools:layout_editor_absoluteY="637dp" />

   TextView
   android:id="@+id/textView"
   android:layout_width="146dp"
   android:layout_height="40dp"
   android:text="TextView"
   tools:layout_editor_absoluteX="54dp"
   tools:layout_editor_absoluteY="275dp" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Output:


UIDemo

Username

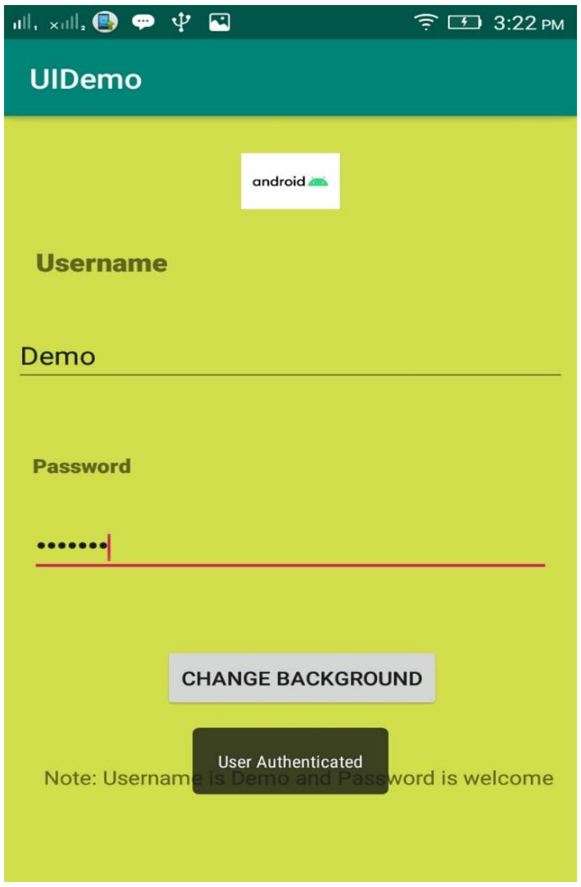
Enter Your Username

Password

Enter Your Password

CHANGE BACKGROUND

Note: Username is Demo and Password is welcome


UIDemo

Username

Demo

Password

•••••••

CHANGE BACKGROUND

User Authenticated

Note: Username is Demo and Password is welcome

Conclusion:

Thus, we have performed the experiment to use the GUI components in the android studio and made an app which shows the color change the text and font type used also different size of the text, background color and successfully executed it.

Experiment No.: 8

Aim: To make an application that draws basic graphical primitives on the screen.

Theory:

The android.graphics.Canvas can be used to draw graphics in android. It provides methods to draw oval, rectangle, picture, text, line etc.

The android.graphics.Paint class is used with canvas to draw objects. It holds the information of color and style.

Android Canvas class encapsulates the bitmaps used as surface. It exposes the draw methods which can be used for designing. Let us first clear the following terms:
Bitmap: The surface being drawn on.
Paint: It lets us specify how to draw the primitives on bitmap. It is also referred to as "Brush".
Canvas: It supplies the draw methods used to draw primitives on underlying bitmap.
Each drawing object specifies a paint object to render. Let us see the available list of drawing objects and they are as follows:
drawArc: This draws an arc between the two angles bounded by an area of rectangle.
drawBitmap: It draws an bitmap on canvas.
drawRGB/drawARGB/drawColor: This fills the canvas with a single color.
drawBitmapMesh: It draws a bitmap using a mesh. It manipulates the appearance of target by moving points on it.
drawCircle: This draws a circle on a specified radius centered on a given point.
drawLine(s):it draws a line (or series of lines) between points.
drawOval: it draws an oval which is bounded by the area of rectangle.
drawPaint: It fills the entire canvas with a specific paint.
drawPath: It draws a path as per specification.
drawPicture: It draws a picture specified on a rectangular area.
drawPosText: it draws a text string specifying the offset of each character.
drawRect: It draws a rectangle.
drawRoundRect: it draws a rectangle with round edges.
drawText: It draws a text string on canvas.
The **Paint** class consists of a paint brush and a palette. It lets us choose how to render the primitives drawn into canvas by draw methods. We can control the color, style, font, special effects etc can be modified by modifying the paint object. For instance, **setColor** method can be used to select the color of Paint. Paint class supports transparency so it can be used to control variety of shades or effects, etc. Let us create a simple example and see the basic usage of canvas and paint.

Code:

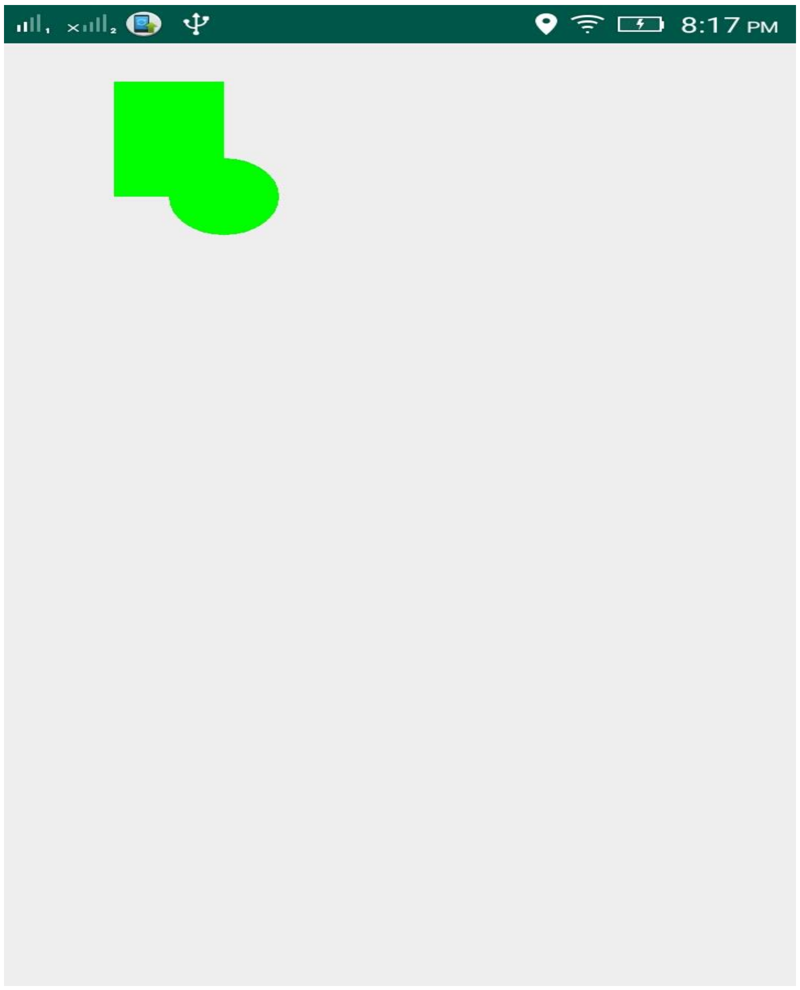
MainActivity.java:

```
package com.example.ex4;

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.os.Bundle;

import com.google.android.material.floatingactionbutton.FloatingActionButton;
import com.google.android.material.snackbar.Snackbar;

import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;

import android.view.View;
import android.view.Menu;
import android.view.MenuItem;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(new myView(this));


    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    private class myView extends View {

        public myView(Context context) {
            super(context);
        }

        @Override
        protected void onDraw(Canvas canvas) {
            super.onDraw(canvas);
            Paint paint=new Paint();
            paint.setColor(Color.GREEN);
            canvas.drawCircle(200,200,50,paint);
            canvas.drawRect(100,50,200,200,paint);
        }
    }
}
```

content_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:context=".MainActivity"
    tools:showIn="@layout/activity_main">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Output:



Conclusion:

Thus, we have performed the experiment to draw basic graphical primitives on the screen in the android app using canvas in android studio and over here we have drawn circle and rectangle as example and successfully executed it.

Experiment No.: 9

Aim: Develop an application that makes use of database

Theory:

SQLite is a opensource SQL database that stores data to a text file on a device. Android comes in with built in SQLite database implementation.

SQLite supports all the relational database features. In order to access this database, you don't need to establish any kind of connections for it like JDBC,ODBC e.t.c

Database - Package

The main package is android.database.sqlite that contains the classes to manage your own databases

Database - Creation

In order to create a database you just need to call this method openOrCreateDatabase with your database name and mode as a parameter.

Database - Insertion

we can create table or insert data into table using execSQL method defined in SQLiteDatabase class.

Database - Fetching

We can retrieve anything from database using an object of the Cursor class. We will call a method of this class called rawQuery and it will return a resultset with the cursor pointing to the table. We can move the cursor forward and retrieve the data.

Database - Helper class

For managing all the operations related to the database , an helper class has been given and is called SQLiteOpenHelper. It automatically manages the creation and update of the database.

Code:

DBHelper.java

```
package com.example.ex5;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

public class DBHelper extends SQLiteOpenHelper {

    public static final String DATABASE_NAME="student.db";
    public DBHelper(Context context){
        super(context,DATABASE_NAME,null,1);
    }
    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL("create table student(sname TEXT,rollno TEXT)");

    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL("DROP TABLE IF EXISTS student");
        onCreate(db);

    }
    public void insertdata(String studname,String studroll){
        SQLiteDatabase db=this.getWritableDatabase();
        ContentValues cv=new ContentValues();
        cv.put("sname",studname);
        cv.put("rollno",studroll);
        db.insert("student",null,cv);
    }

    public Cursor getalldata(){
        SQLiteDatabase db=this.getWritableDatabase();
        Cursor res=db.rawQuery("select * from student",null);
        return res;
    }
}
```

content_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
```

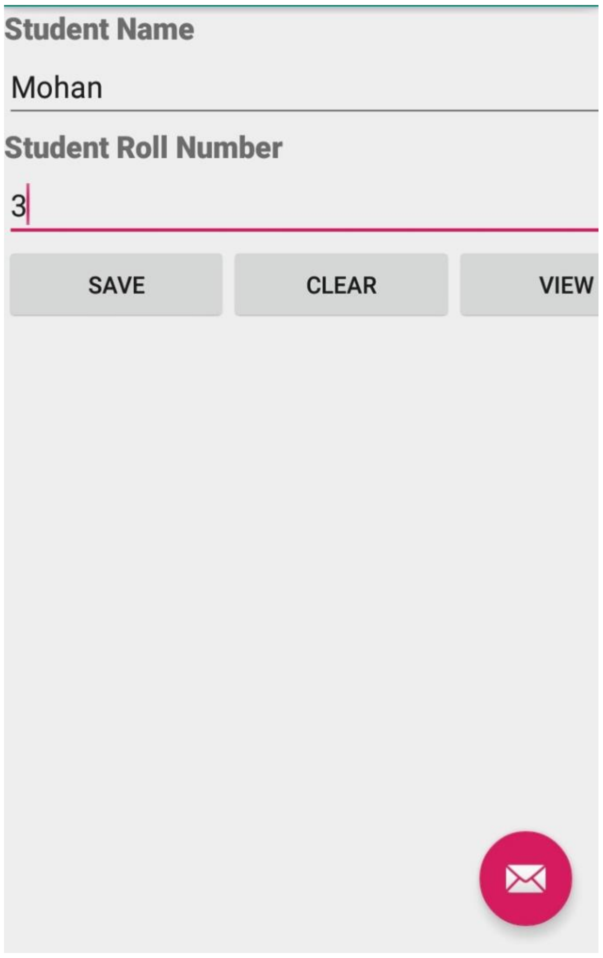
```
android:layout_width="match_parent"
android:layout_height="match_parent"
app:layout_behavior="@string/appbar_scrolling_view_behavior"
tools:context=".MainActivity"
tools:showIn="@layout/activity_main">

<LinearLayout
  android:layout_width="409dp"
  android:layout_height="673dp"
  android:orientation="vertical"
  tools:layout_editor_absoluteX="1dp"
  tools:layout_editor_absoluteY="1dp">

  <TextView
    android:id="@+id/textView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:fontFamily="sans-serif-black"
    android:text="Student Name"
    android:textSize="18sp" />

  <EditText
    android:id="@+id/studname"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:ems="10"
    android:inputType="textPersonName" />

  <TextView
    android:id="@+id/textView2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:fontFamily="sans-serif-black"
    android:text="Student Roll Number"
    android:textSize="18sp" />

  <EditText
    android:id="@+id/studrollno"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:ems="10"
    android:inputType="textPersonName" />

  <LinearLayout
```

```
        android:layout_width="match_parent"
        android:layout_height="192dp"
        android:orientation="horizontal">

        <Button
            android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Save"
            android:onClick="insert"/>

        <Button
            android:id="@+id/button2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Clear"
            android:onClick="clear"/>

        <Button
            android:id="@+id/button3"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="View"
            android:onClick="viewdata"/>
    </LinearLayout>
  </LinearLayout>

</androidx.constraintlayout.widget.ConstraintLayout>
```

MainActivity.java

```
package com.example.ex5;

import android.database.Cursor;
import android.os.Bundle;

import com.google.android.material.floatingactionbutton.FloatingActionButton;
import com.google.android.material.snackbar.Snackbar;

import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;

import android.view.View;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    EditText E1,E2;
  DBHelper mydb;
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Toolbar toolbar = findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);

    FloatingActionButton fab = findViewById(R.id.fab);
    fab.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View view) {
        Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
            .setAction("Action", null).show();
      }
    });

    mydb=new DBHelper(this);
    E1=(EditText)findViewById(R.id.studname);
    E2=(EditText)findViewById(R.id.studrollno);
  }

  @Override
  public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true;
  }

  @Override
  public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();

    //noinspection SimplifiableIfStatement
    if (id == R.id.action_settings) {
        return true;
    }

    return super.onOptionsItemSelected(item);
}

public void insert(View view) {
    mydb.insertdata(E1.getText().toString(),E2.getText().toString());
    Toast.makeText(this,"Data Inserted",Toast.LENGTH_LONG).show();
}

public void clear(View view) {
    E1.setText("");
    E2.setText("");
}

public void viewdata(View view) {
    Cursor c=mydb.getalldata();
    if(c.getCount()==0){
        showdialog("Alert","No Data Found");
    }
    else {
        StringBuffer bf=new StringBuffer();
        while (c.moveToNext()){
            bf.append("Name:"+c.getString(0)+"\n");
            bf.append("Roll No:"+c.getString(1)+"\n");
        }
        showdialog("Data",bf.toString());
    }

}

public void showdialog(String title,String msg){
    AlertDialog.Builder bu=new AlertDialog.Builder(this);
    bu.setCancelable(true);
    bu.setTitle(title);
    bu.setMessage(msg);
    bu.show();}}
```

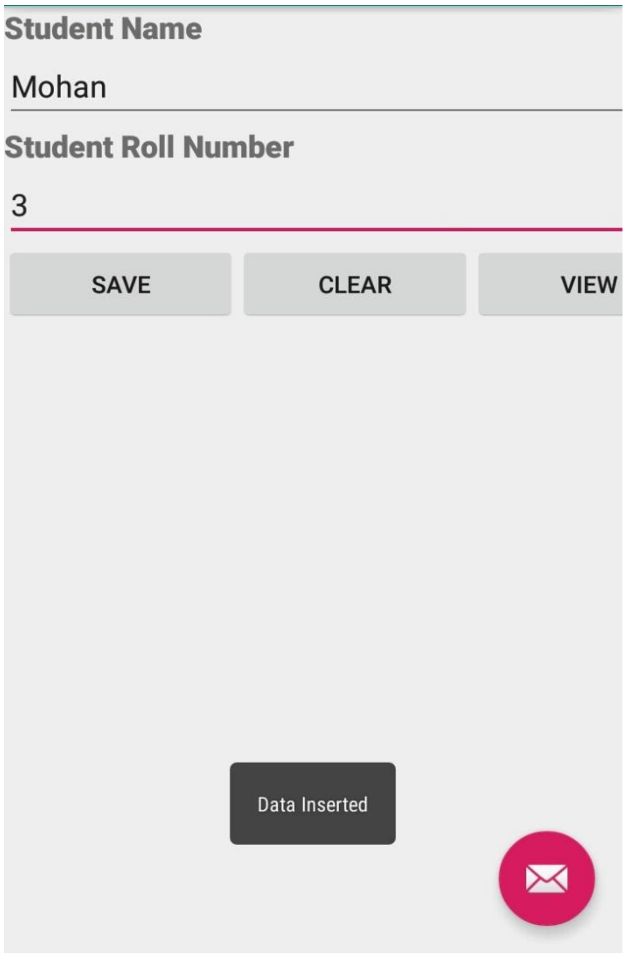
Output:


Student Name

Mohan

Student Roll Number

3

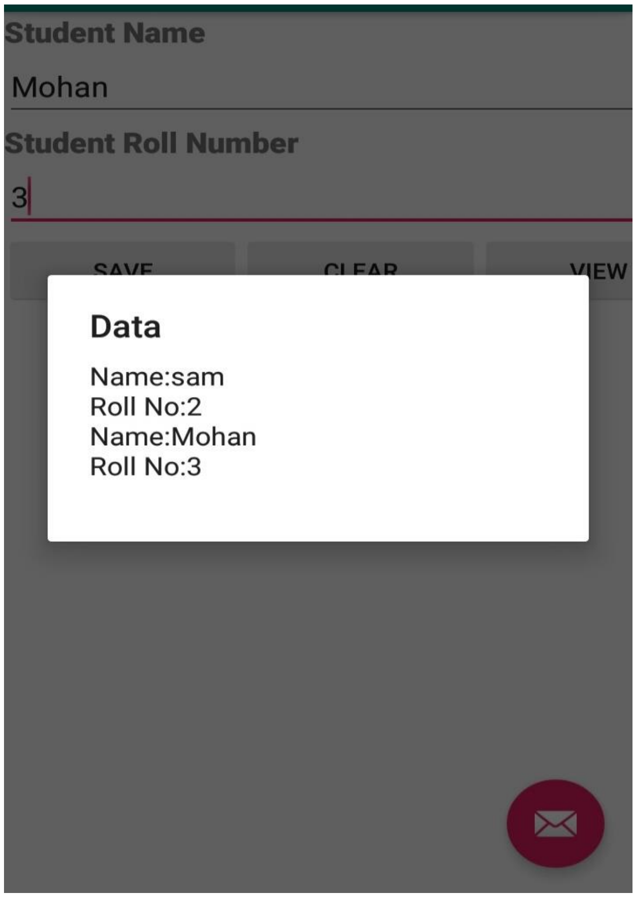

Student Name

Mohan

Student Roll Number

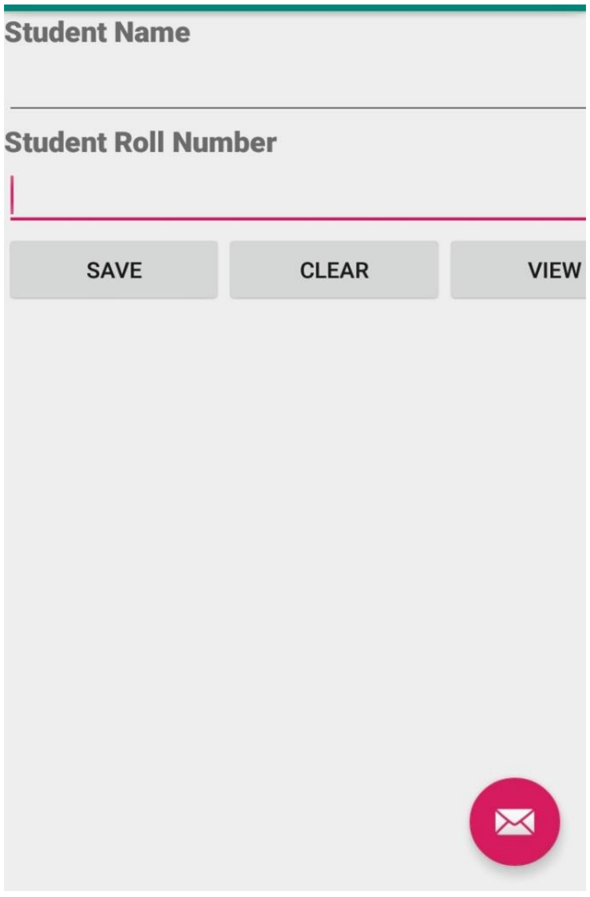
3

Data Inserted



Data

Name:sam
Roll No:2
Name:Mohan
Roll No:3



Conclusion:

Thus, we have performed the experiment and made an app that uses SQLite database over here takes the student name and roll number and input and stores that in database which can be viewed later. We have successfully executed the experiment.

Experiment No.: 10

Aim: Implement an application that creates an alert upon receiving a message.

Theory:

A notification is a message you can display to the user outside of your application's normal UI. When you tell the system to issue a notification, it first appears as an icon in the notification area. To see the details of the notification, the user opens the notification drawer. Both the notification area and the notification drawer are system-controlled areas that the user can view at any time.

Android **Toast** class provides a handy way to show users alerts but problem is that these alerts are not persistent which means alert flashes on the screen for a few seconds and then disappears.

To see the details of the notification, you will have to select the icon which will display notification drawer having detail about the notification. While working with emulator with virtual device, you will have to click and drag down the status bar to expand it which will give you detail as follows. This will be just **64 dp** tall and called normal view.

Create and Send Notifications

You have simple way to create a notification. Follow the following steps in your application to create a notification –

Step 1 - Create Notification Builder

As a first step is to create a notification builder using *NotificationCompat.Builder.build()*. You will use Notification Builder to set various Notification properties like its small and large icons, title, priority etc.

Step 2 - Setting Notification Properties

Once you have **Builder** object, you can set its Notification properties using Builder object as per your requirement. But this is mandatory to set at least following −

- A small icon, set by **setSmallIcon()**
- A title, set by **setContentTitle()**
- Detail text, set by **setContentText()**

Step 3 - Attach Actions

This is an optional part and required if you want to attach an action with the notification. An action allows users to go directly from the notification to an **Activity** in your application, where they can look at one or more events or do further work.

Step 4 - Issue the notification

Finally, you pass the Notification object to the system by calling NotificationManager.notify() to send your notification. Make sure you call **NotificationCompat.Builder.build()** method on builder object before notifying it. This method combines all of the options that have been set and return a new **Notification** object.

Code:

MainActivity.java

```
package com.example.ex10;

import android.app.Notification;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Intent;
import android.os.Bundle;

import com.google.android.material.floatingactionbutton.FloatingActionButton;
import com.google.android.material.snackbar.Snackbar;

import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;

import android.view.View;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.EditText;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        FloatingActionButton fab = findViewById(R.id.fab);
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
                        .setAction("Action", null).show();
            }
        });
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    public void notify(View view) {
        EditText E1=(EditText)findViewById(R.id.editText);
        Intent intent=new Intent(this,ResultActivity.class);
        PendingIntent pending=PendingIntent.getActivity(this,0,intent,0);
        Notification noti=new Notification.Builder(this)
                .setContentTitle("New Message")
                .setContentText(E1.getText().toString()).setSmallIcon(R.mipmap.ic_launcher)
                .setContentIntent(pending).build();
        NotificationManager
manager=(NotificationManager)getSystemService(NOTIFICATION_SERVICE);
        noti.flags |=Notification.FLAG_AUTO_CANCEL;
        manager.notify(0,noti);

    }
}
```

content_result.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   app:layout_behavior="@string/appbar_scrolling_view_behavior"
   tools:context=".ResultActivity"
   tools:showIn="@layout/activity_result">

</androidx.constraintlayout.widget.ConstraintLayout>
```

content_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   app:layout_behavior="@string/appbar_scrolling_view_behavior"
   tools:context=".MainActivity"
   tools:showIn="@layout/activity_main">

   <LinearLayout
      android:layout_width="409dp"
      android:layout_height="673dp"
      android:orientation="vertical"
      tools:layout_editor_absoluteX="1dp"
      tools:layout_editor_absoluteY="1dp">

      <TextView
         android:id="@+id/textView"
         android:layout_width="match_parent"
         android:layout_height="wrap_content"
         android:fontFamily="sans-serif-black"
         android:text="Message"
         android:textSize="24sp" />

      <EditText
```

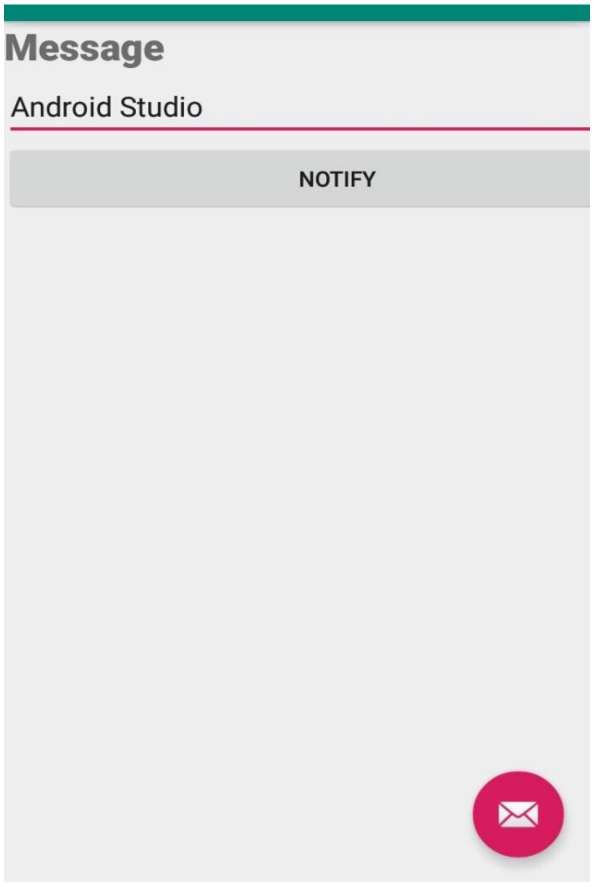
```
        android:id="@+id/editText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:gravity="start|top"
        android:inputType="textMultiLine" />

    <Button
        android:id="@+id/button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Notify"
        android:onClick="notify"/>
  </LinearLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```

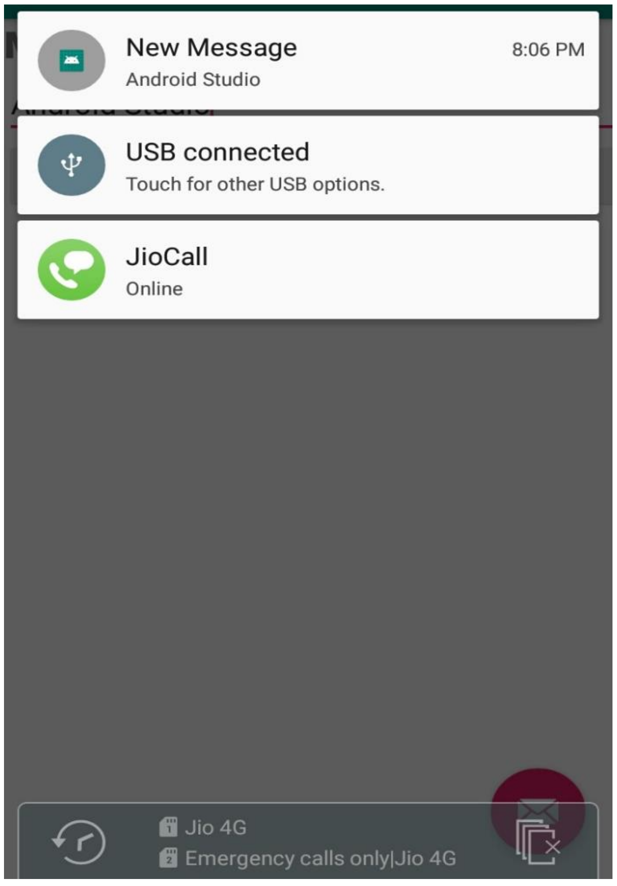
ResultActivity.java

```
package com.example.ex10;

import android.os.Bundle;

import com.google.android.material.floatingactionbutton.FloatingActionButton;
import com.google.android.material.snackbar.Snackbar;

import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;

import android.view.View;

public class ResultActivity extends AppCompatActivity {

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_result);
    Toolbar toolbar = findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);

    FloatingActionButton fab = findViewById(R.id.fab);
    fab.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View view) {
```

```
                Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
                        .setAction("Action", null).show();
            }
        });
    }

}
```

Output:




New Message 8:06 PM
Android Studio

USB connected
Touch for other USB options.

JioCall
Online

Jio 4G
Emergency calls only|Jio 4G

Conclusion:

Thus, we have performed the experiment in which we have developed an app where a alert is created when a message is sent and we have executed it properly.