

MOBILE COMPUTATION

Practical File



Akjot Singh

CSE-1

00213202717

Experiment -1

Aim: Write a program to implement a simple calculator using text view, edit view, option button and button._

Procedure: 1. Open Android Studio and Create a new project

2. Choose Empty Activity for this Project.

3. The names for activity are default Main Activity and keep all the things by default and Click Finish.

4. Design the calculator using edit text, option button and radio button in XML.

5. Using if condition the arithmetic operation is performed.

6. Assign an object to display result.

PROGRAM:

Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<LinearLayout
```

```
xmlns:android="http://schemas.android.com/apk/res/android"
```

```
xmlns:tools="http://schemas.android.com/tools"
```

```
android:layout_width="match_parent"
```

```
android:layout_height="match_parent"
```

```
android:paddingLeft="@dimen/activity_horizontal_margin"
```

```
android:paddingRight="@dimen/activity_horizontal_margin"
```

```
android:paddingTop="@dimen/activity_vertical_margin"
```

```
android:orientation="vertical"
```

```
android:paddingBottom="@dimen/activity_vertical_margin"
tools:context=".MainActivity">
<TextView
android:id="@+id/textview1"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="First Number"
android:textSize="20dp"/>
<EditText
android:id="@+id/firstNum"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:ems="10"
android:inputType="number"
android:textSize="20dp"/>
<TextView
android:id="@+id/textview2"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="Second Number"
android:textSize="20dp"/>
```

```
<EditText
android:id="@+id/secondNum"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:ems="10"
android:inputType="number"
android:textSize="20dp"/>
<RadioGroup
android:layout_width="match_parent"
android:layout_height="173dp">
<RadioButton
android:id="@+id/addno"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_weight="1"
android:textSize="20dp"
android:text="Add" />
<RadioButton
android:id="@+id/subno"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
```

```
android:layout_weight="1"
android:textSize="20dp"
android:text="Subtract" />
<RadioButton
android:id="@+id/mulno"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_weight="1"
android:textSize="20dp"
android:text="Multiply" />
<RadioButton
android:id="@+id/divno"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_weight="1"
android:textSize="20dp"
android:text="Divide" />
<TextView
android:id="@+id/result"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
```

```
android:layout_weight="1"
android:textSize="20dp"
android:text="Result " />
</RadioGroup>
<Button
android:id="@+id/calculate"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="Calculate"
android:onClick="calculateButtonClick"/>
</LinearLayout>
```

MainActivity.java

```
package com.example.malathi.sample;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
    EditText first,second;
```

```
RadioButton add,sub,mul,div;
```

```
@Override
```

```
protected void onCreate(Bundle savedInstanceState) {
```

```
    super.onCreate(savedInstanceState);
```

```
    setContentView(R.layout.activity_main);
```

```
}
```

```
public void calculateButtonClick(View v)
```

```
{
```

```
    first=(EditText)findViewById (R.id.firstNum);
```

```
    second=(EditText)findViewById (R.id.secondNum);
```

```
    float firstNo = Integer.parseInt(first.getText().toString());
```

```
    float secondNo = Integer.parseInt(second.getText().toString());
```

```
    add =(RadioButton) findViewById(R.id.addno);
```

```
    sub =(RadioButton) findViewById(R.id.subno);
```

```
    mul =(RadioButton) findViewById(R.id.mulno);
```

```
    div =(RadioButton) findViewById(R.id.divno);
```

```
    float result = 0;
```

```
    if(add.isChecked())
```

```
{
```

```
        result = firstNo + secondNo;
```

```
    } else if (sub.isChecked()) {
```



```

result = firstNo - secondNo;
} else if (mul.isChecked()){
result = firstNo * secondNo;
} else if (div.isChecked()){
result = firstNo / secondNo;
}
TextView res = (TextView)findViewById(R.id.result);
res.setText("Result : " +result);
}
}

```

RESULT:

Thus the application of simple calculator using edit text, option button and radio button was created and verified successfully.

Addition

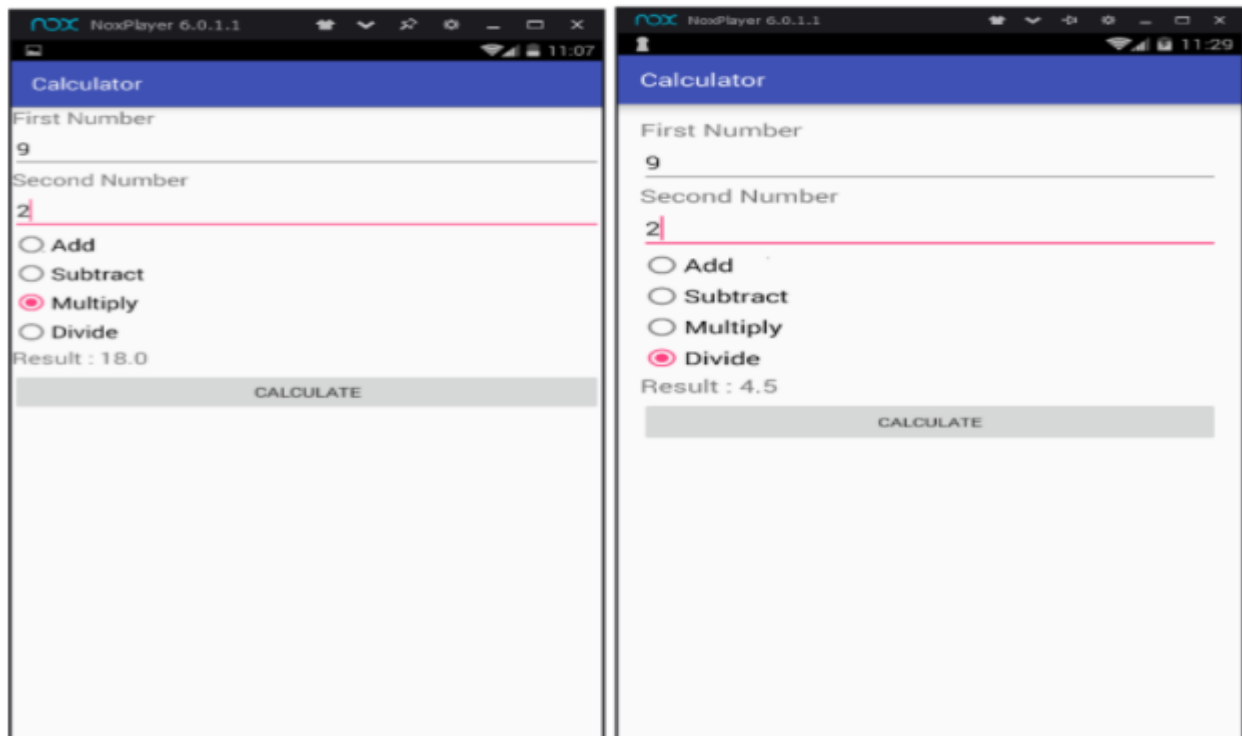


Subtraction



Multiplication

Divide



Experiment -2

AIM: Write a program to demonstrate a photo gallery.

- Procedure:**
1. Create a new project and name it Gallery.
 2. The images to display are placed in res ⇒drawable-hdpi folder.
 3. To create an Adapter class which extends BaseAdapter class and override getView() method
 4. getCount()method returns the total number of items to be displayed in a list.
 5. getView() method called automatically for all items of Gallery

PROGRAM:

Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
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"
android:orientation="vertical"
tools:context="com.example.malathi.gallery.MainActivity">
<Gallery
android:id="@+id/gallery"
android:layout_width="fill_parent"
android:layout_height="wrap_content"/>
<ImageView
android:id="@+id/imgGalleryImage"
android:layout_width="fill_parent"
android:layout_height="fill_parent"/>
</LinearLayout>

```

MainActivity.java

```

package com.example.malathi.gallery;
import android.content.Context;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.BaseAdapter;
import android.widget.Gallery;
import android.widget.ImageView;
public class MainActivity extends AppCompatActivity {
    Gallery Imagegallery;
    Integer[] GalleryImagesList=
    {
        R.drawable.one,R.drawable.two,
        R.drawable.three,R.drawable.four,
        R.drawable.five,R.drawable.six,

```

```

R.drawable.seven
};
ImageViewimgGalleryImage;
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
imgGalleryImage= (ImageView)findViewById(R.id.imgGalleryImage);
imgGalleryImage.setImageResource(R.drawable.one);
Imagegallery= (Gallery)findViewById(R.id.gallery);
Imagegallery.setAdapter(new ImageAdapter(this));
Imagegallery.setOnItemClickListener(new
AdapterView.OnItemClickListener()
{
@Override
public void onItemClick(AdapterView parent, View view,intposition, long
id)
{
imgGalleryImage.setImageResource(GalleryImagesList[position]);
}
});
}
private class ImageAdapterextends BaseAdapter
{
Context context;
public ImageAdapter(Context context)
{
this.context= context;
}
@Override
public intgetCount()
{

```

```

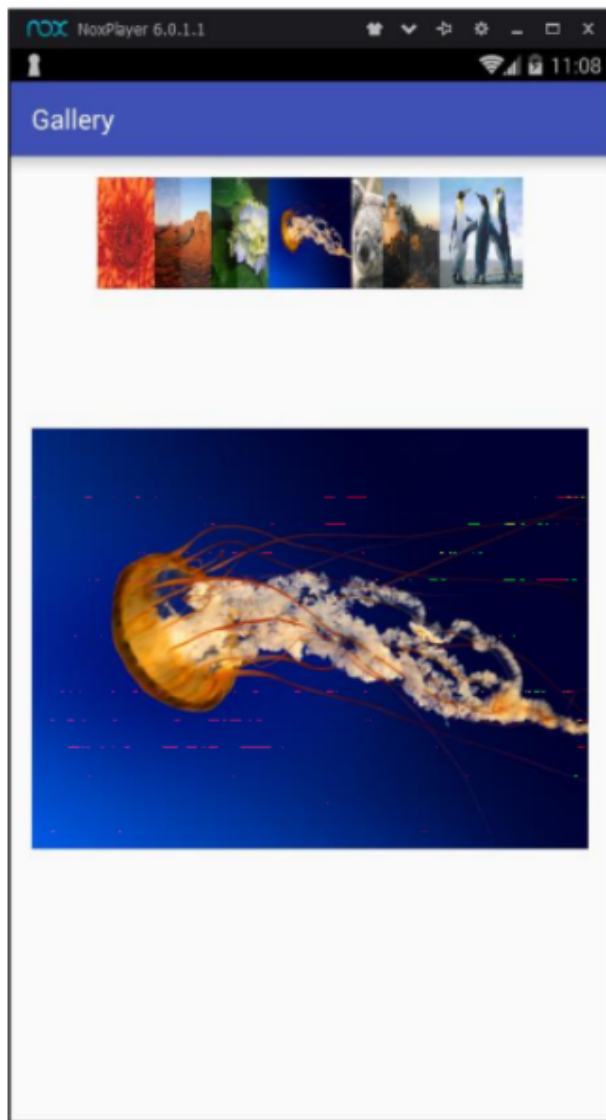
return GalleryImagesList.length;
}
@Override
public Object getItem(intposition)
{
return GalleryImagesList[position];
}
@Override
public long getItemId(intposition)
{
return position;
}
@Override
public View getView(intposition, View convertView, ViewGroup parent)
{
ImageViewimageView = new ImageView(this.context);
imageView.setImageResource(GalleryImagesList[position]);
imageView.setLayoutParams(new Gallery.LayoutParams(150, 200));
imageView.setScaleType(ImageView.ScaleType.FIT_XY);
return imageView;
}
}
}

```

RESULT:

Hence the program to demonstrate photo gallery was executed and verified successfully.

Output:



Experiment -3

AIM: Write a program to demonstrate Date picker and time picker.

Procedure: 1. Create a new project and name it datetimepicker.
2. Choose Empty activity.
3. Drag timepicker,datepicker and buttons in xml design part.
4. In java declare the objects for timepicker, datepicker and also variables for year, month and day.
5. Obtain timepicker, datepicker attributes from layout.
6. Display date and time picker value as pop up notification using Toast.

PROGRAM:

Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
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"
tools:context="com.example.malathi.datetimepicker.MainActivity"
"
android:orientation="vertical">
<LinearLayout android:layout_width="match_parent"
android:layout_height="wrap_content"
android:orientation="vertical">
```

```
<DatePicker
android:id="@+id/datePicker"
android:layout_width="wrap_content"
android:layout_height="wrap_content " />
<Button android:id="@+id/datebutton"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Display date" />
<TimePicker android:id="@+id/timePicker"
android:layout_width="wrap_content "
android:layout_height="wrap_content " />
<Button android:id="@+id/timebutton"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Verify Time" />
</LinearLayout>
</ScrollView>
```

MainActivity.java

```
package com.example.malathi.datetimepicker;
import android.app.DatePickerDialog;
import android.app.TimePickerDialog;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.TimePicker;
import android.widget.Toast;
```



```

public class MainActivity extends AppCompatActivity {
    DatePicker datepicker;
    TimePicker timepicker;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        datepicker= (DatePicker) findViewById(R.id.datePicker);
        timepicker= (TimePicker) findViewById(R.id.timePicker);
        //timepicker.setIs24HourView(true);
        Button button = (Button) findViewById(R.id.datebutton);
        Button button1 = (Button) findViewById(R.id.timebutton);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Toast.makeText(getApplicationContext(), "Date : "
                    + (datepicker.getMonth() + 1)
                    + "/" + datepicker.getDayOfMonth() + "/" + datepicker.getYear(),
                    Toast.LENGTH_LONG).show(); }
        });
        button1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Toast.makeText(getApplicationContext(),"Time Selected :
                    "+timepicker.getCurrentHour()
                    +":"+timepicker.getCurrentMinute(),
                    Toast.LENGTH_SHORT).show();
            }
        }); }

```

```
}
```

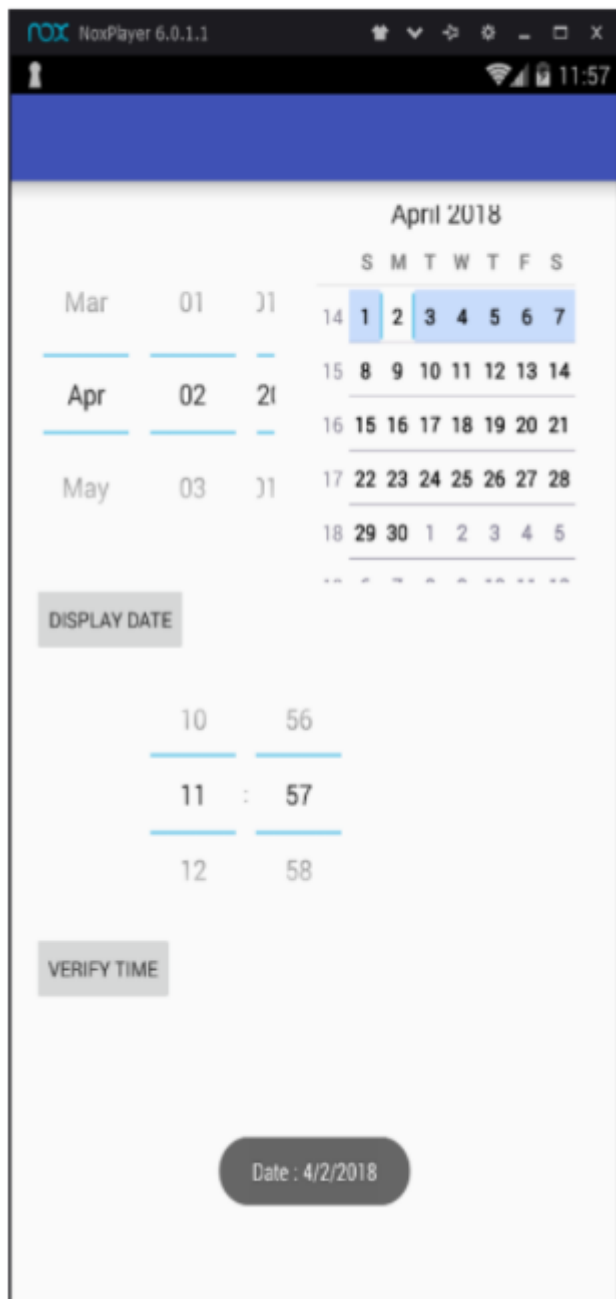
RESULT:

Hence the program to demonstrate date picker and time picker was executed and verified successfully.

Output:

Date Picker

Time Picker



Experiment - 4

AIM: Develop an application to send SMS.

Procedure:

1. Create a project
2. Drag the two edittext, two textview and one button from pallet. (activity_main.xml)
3. Add permission code in AndroidManifest to send SMS
4. Get the smsmanger instance and call the sendTextMessage method to send message.

PROGRAM:

AndroidManifest.xml

```
<uses-permission android:name="android.permission.SEND_SMS"/>
```

Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout 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"
    android:orientation="vertical"
    tools:context="com.example.malathi.sms.MainActivity">

    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

```
android:text="Phone Number" />

<EditText

android:id="@+id/editTextnumber"

android:layout_width="match_parent"

android:layout_height="wrap_content"

android:ems="10"

android:inputType="phone" />

<TextView

android:id="@+id/textView2"

android:layout_width="match_parent"

android:layout_height="wrap_content"

android:text="Message" />

<EditText

android:id="@+id/editText2msg"

android:layout_width="match_parent"

android:layout_height="wrap_content"

android:ems="10"

android:inputType="textMultiLine" />

<Button

android:id="@+id/button"

android:layout_width="match_parent"

android:layout_height="wrap_content"

android:text="send"/>

</LinearLayout>
```

MainActivity.java

```
package com.example.malathi.sms;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.telephony.SmsManager;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    EditText no;

    EditText message;

    Button send;

    @Override

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        no = (EditText) findViewById(R.id.editTextnumber);
        message = (EditText) findViewById(R.id.editText2msg);
        send = (Button) findViewById(R.id.button);
        send.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String number = no.getText().toString();
                String msg = message.getText().toString();
                try {
                    SmsManager sms = SmsManager.getDefault();
```

```
sms.sendTextMessage(number,null,msg,null,null);  
Toast.makeText(getApplicationContext(), "send",  
Toast.LENGTH_LONG).show();  
  
catch (Exception e) {  
    Toast.makeText(getApplicationContext(), "not send",  
    Toast.LENGTH_LONG).show();  
    e.printStackTrace();  
}  
}  
});  
}  
}
```

RESULT:

Thus SMS sending application was developed successfully.

Output:



Experiment - 5

AIM: Write a program to view, edit contact.

Procedure:

1. Create a project
2. Drag EditText, Button in activity_main.xml
3. Add READ_CONTACT permission in androidmanifest.xml.
4. EditContact() method is used to open links clicked by the user.

PROGRAM:

AndroidManifest.xml

```
<uses-permission
```

```
android:name="android.permission.READ_CONTACTS"/>
```

Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<LinearLayout 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"
```

```
android:orientation="vertical"
```

```
tools:context="com.example.malathi.example.MainActivity">
```

```
<Button
```

```
android:layout_gravity="center"
```

```
android:layout_width="wrap_content"
```

```
android:text="view contact"
```

```
android:layout_height="wrap_content"
```

```
android:id="@+id/button"/>
<EditText
android:id="@+id/editText"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="Phone Number"
android:ems="10"
android:inputType="phone" />
<Button
android:layout_gravity="center"
android:id="@+id/button2"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="edit contact" />
</LinearLayout>
```

MainActivity.java

```
package com.example.malathi.example;

import android.content.ContentUris;
import android.content.Intent;
import android.database.Cursor;
import android.net.Uri;
import android.provider.ContactsContract;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
```

```
import android.widget.EditText;

import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    Button view,edit;

    EditText num;

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);

        view=(Button)findViewById(R.id.button);

        edit=(Button)findViewById(R.id.button2);

        num=(EditText)findViewById(R.id.editText);

        view.setOnClickListener(new View.OnClickListener() {

            @Override

            public void onClick(View view) {

                Intent read1=new Intent();

                read1.setAction(android.content.Intent.ACTION_VIEW);

                read1.setData(ContactsContract.Contacts.CONTENT_URI);

                startActivity(read1);

            }

        });

        edit.setOnClickListener(new View.OnClickListener() {

            @Override

            public void onClick(View view) {

                EditContact(num.getText().toString());

            }

        });

    }

}
```

```
});  
  
}  
  
private void EditContact(String contactNumber) {  
  
    Uri lookupUri =  
  
    Uri.withAppendedPath(ContactsContract.Contacts.CONTENT_FILTER_URI, Uri.encode(contactNumber));  
  
    Cursor mcursor = getContentResolver().query(lookupUri, null, null, null, null);  
  
    long idPhone = 0;  
  
    try {  
  
        if (mcursor != null) {  
  
            if (mcursor.moveToFirst()) {  
  
                idPhone =  
  
                Long.valueOf(mcursor.getString(mcursor.getColumnIndex(ContactsContract.  
                PhoneLookup._ID)));  
  
                Intent editContact = new Intent(Intent.ACTION_EDIT);  
  
                editContact.setData(ContentUris.withAppendedId(ContactsContract.Contacts.  
                CONTENT_URI, idPhone));  
  
                startActivity(editContact);  
  
            } else  
  
                Toast.makeText(MainActivity.this, "Contact not found",  
                Toast.LENGTH_SHORT).show();  
  
            }  
  
        } finally {  
  
            mcursor.close();  
  
        }  
  
    }  
  
}
```

}

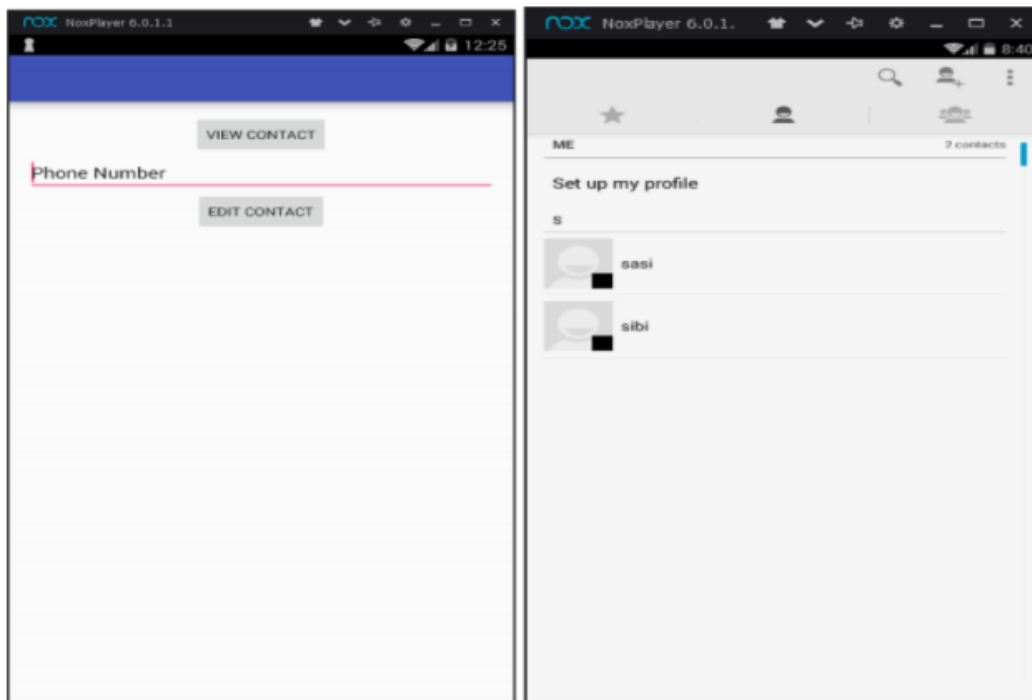
}

RESULT:

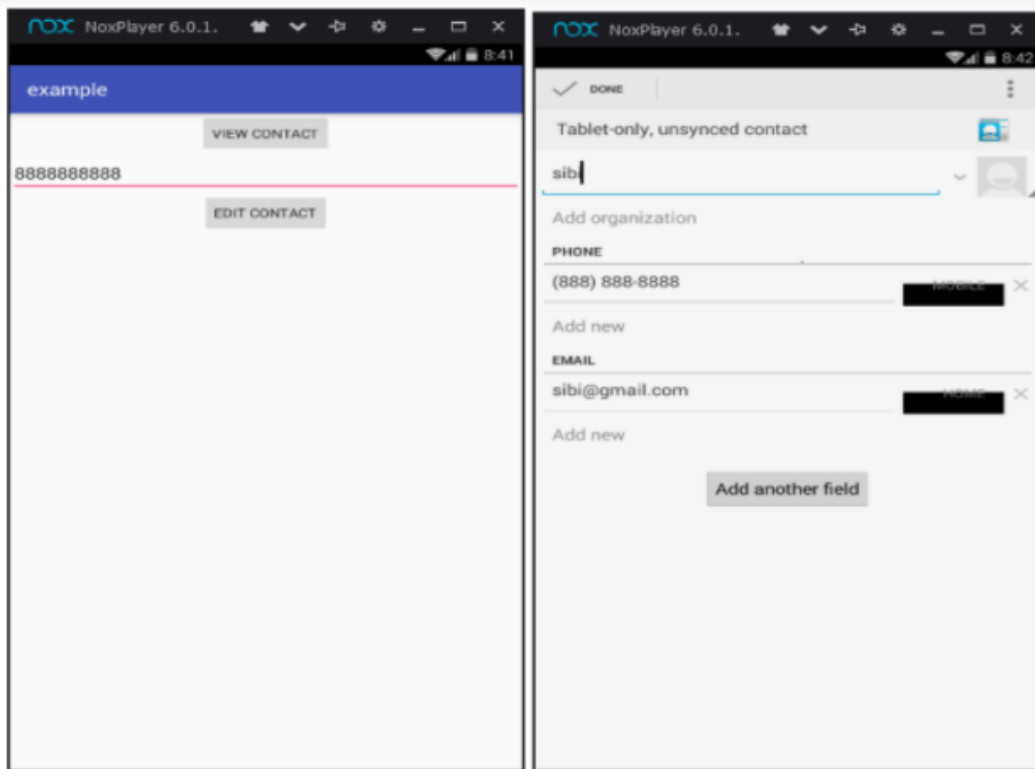
Thus the program to view and edit contact was executed and verified successfully.

OUTPUT:

View Contact



Edit Contact



Experiment - 6

AIM: Write a program to send e-mail.

Procedure:

1. Create a new project.
2. Drag the 3 Edittext, 3 Textview and 1 button from the palette in activity_main.xml
3. Intent.ACTION_SEND to call an existing email client to send an Email.

PROGRAM:

Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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"
android:orientation="vertical"
tools:context="com.example.malathi.email.MainActivity">
<TextView
android:id="@+id/textView"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="To" />
<EditText
android:id="@+id/editTextmail"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:ems="10"
android:inputType="textEmailAddress" />
<TextView android:id="@+id/textView2"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="Subject" />
<EditText android:id="@+id/editText2sub"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:ems="10"
android:inputType="textPersonName" />
<TextView
android:id="@+id/textView3"
android:layout_width="match_parent"
android:layout_height="wrap_content"
```

```
android:text="Message" />
<EditText
    android:id="@+id/editText3msg"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:ems="10"
    android:inputType="textMultiLine" />
<Button
    android:id="@+id/button"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Send" />
</LinearLayout>
```

MainActivity.java

```
Package com.example.malathi.email;
import android.content.Intent;
import android.os.Message;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
public class MainActivity extends AppCompatActivity {
    EditText to;
    EditText subject;
    EditText message;
    Button send;
```



```

@Override

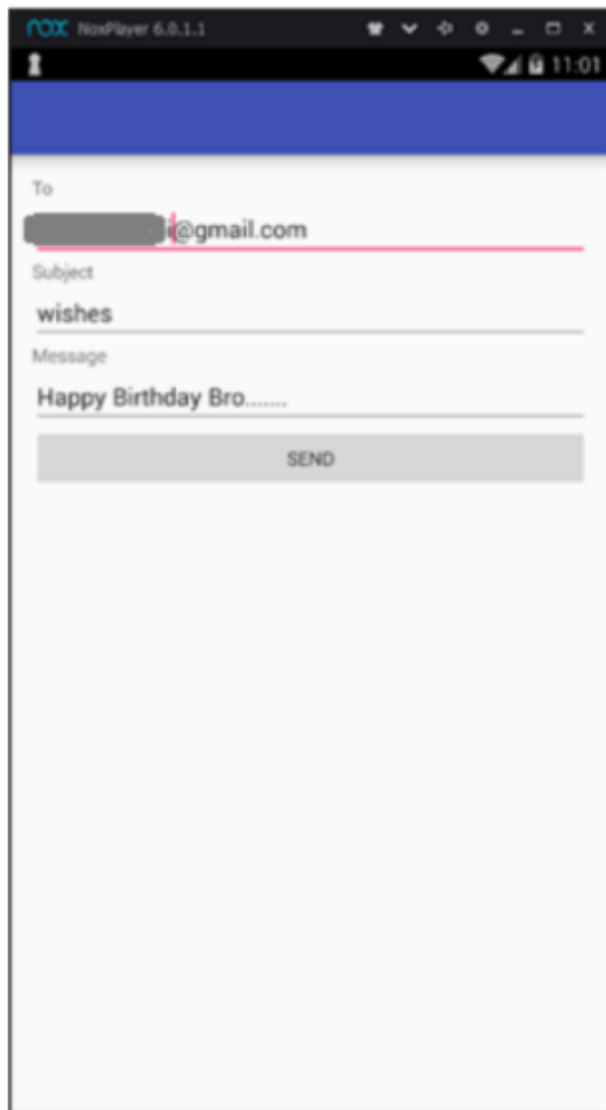
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    to=(EditText)findViewById(R.id.editTextmail);
    subject=(EditText)findViewById(R.id.editText2sub);
    message=(EditText)findViewById(R.id.editText3msg);
    send=(Button)findViewById(R.id.button);
    send.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            String textto=to.getText().toString();
            String textsub=subject.getText().toString();
            String textmsg=message.getText().toString();
            Intent email=new Intent(Intent.ACTION_SEND);
            email.putExtra(Intent.EXTRA_EMAIL,new String[]{textto});
            email.putExtra(Intent.EXTRA_SUBJECT,textsub);
            email.putExtra(Intent.EXTRA_TEXT,textmsg);
            email.setType("text/plain");startActivity(Intent.createChooser(email,"choose
            email:"));
        }
    });
}
}
}

```

RESULT:

Thus the program to send e-mail was executed and verified successfully.

OUTPUT:



Experiment - 7

AIM: To create scenario and study the performance of token bus protocol through simulation.

Introduction: The purpose of this experiment is to understand the concept of demand assignment versus random access, setting priorities and token management in a ring and bus LAN. In this lab you will be able to implement a token-passing access method for a ring and bus LAN.

Hardware Requirement

- 3PCs with NIU card
- Network Emulation Unit
- Jumper Cables

Background


Token bus is a network implementing the token ring protocol over a "virtual ring" on a coaxial cable. A token is passed around the network nodes and only the node possessing the token may transmit. If a node doesn't have anything to send, the token is passed on to the next node on the virtual ring. Each node must know the address of its neighbor in the ring, so a special protocol is needed to notify the other nodes of connections to, and disconnections from, the ring.


Design

Design a ring and bus topology with 4 nodes using token passing mechanism and compare the throughput. Set the lowest priority node with My Address 3 and token holding time of 10000 ms. Ensure Bit delay of 0 seconds is set at NEU.

Procedure

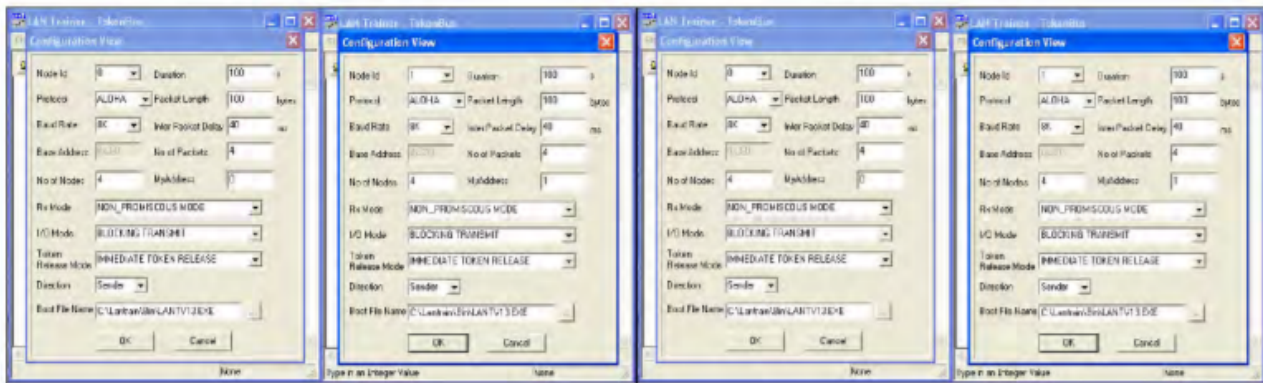
Token Bus

1. Click on the Token Bus icon  twice from the desktop.

2. Click the  Configuration button in the window in both the PC's.

PC 1 – Sender

PC – 2 Sender



Setting the Configurations Menu for Token Bus

PC 1		PC 2	
Node id	0 on config menu 1 and 1 on config menu 2	Node id	0 on config menu 1 and 1 on config menu 2
Protocol	ALOHA	Protocol	ALOHA
Baud Rate	8Kbps (At both the config menu and NEU)	Baud Rate	8Kbps (At both the config menu and NEU)
Duration	100s	Duration	100s
Packet Length	100 bytes	Packet Length	100 bytes
My Address	0 on config menu 1 and 1 on menu 2	My Address	2 on config menu 1 and 3 on menu 2
Bit Delay	0(at NEU)	Bit Delay	0(at NEU)
Direction	Sender	Direction	Sender

- 1.If you connect two PC's and configured four nodes then set the My Address as 0 to 3 in all four nodes, if you connect three PCs and configured six nodes then set the My Address as 0 to 5 in all six nodes.
2. Start running the experiment from the lowest priority node (i.e., from My Address 3 in case of four nodes and 5 in the case of six nodes)
3. No of Nodes has to be set as 4 when two PCs are connected and 6 when three PCs are connected.

$$G = \frac{N * P}{C * t_a} \rightarrow \text{Equation A}$$

G is the generated load in the network.

N is the number of nodes participating in the network. For example, let us say that 4 nodes (using 2 computers)

P is the packet length expressed in bits; say 100 bytes (800 bits). C is the data rate normally set as 8kbs, which is selected in the NEU.

ta is the inter packet delay expressed in seconds; the time interval between two consecutive packets generated.

So, let's assume $t_a = 40$ milliseconds and substitute the above mentioned parameters in the Equation A which leads to $G = 10$. Like wise assume various values of t_a to generate offer loads in the range of 0.1 to 10. Substitute the value of t_a in the configuration menu.

3. Click OK button and Download the driver to the NIU using the BOOT button



command.

Booting from any one of the applications is enough.



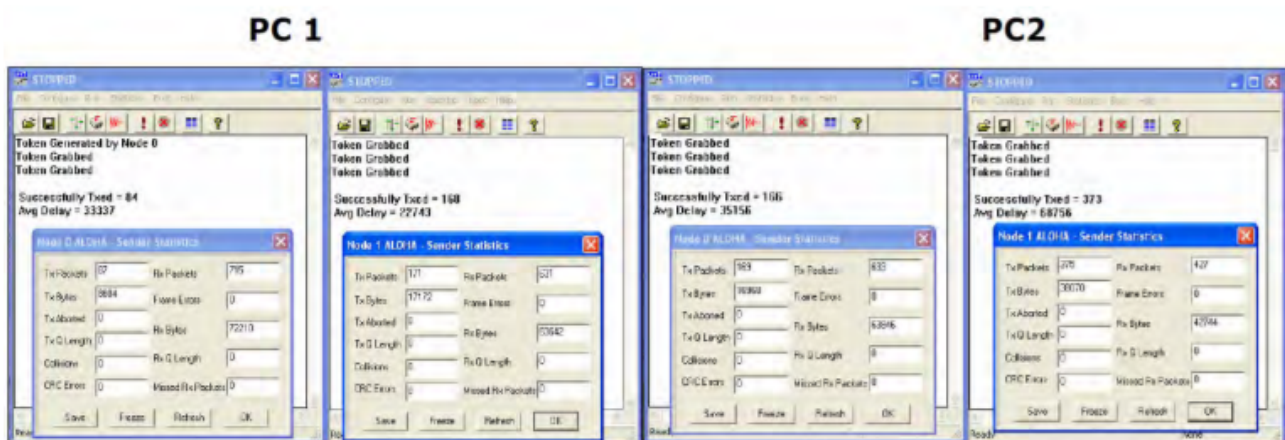
4. Run the experiment by clicking button or by choosing RUN _ Start from each application.

Run the all the experiments at the same time.

Note: While you do this THT window pops up, enter the THT time in all nodes and press the OK button first in the node, which has the lowest priority of My Address.

5. Set the Token Holding Time (THT) (say 10000 ms).

6. View the statistics window for results. To view the statistics window click on button.



7. Note down the readings once the experiment is completed.

8. Repeat the above steps for various values of t_a

9. Calculate the Practical offered load from the below given formula and plot the graph between the practical Offered load and Throughput. Note: You can also use the template for plotting the graph. Please refer Appendix-1 to plot the graph using the template.

10. Repeat the experiment for various values of Packet length, Node, Data rate.

11. Repeat the above steps, while running the experiment set the BER to 10⁻² in the NEU or try to stop one of the nodes and observe the behavior and explain the same.

Calculations of Practical Throughput from the obtained readings

$$X = \frac{(\text{Sum of Tx packet in all the nodes} * \text{Packet Length} * 8)}{(\text{Duration of Experiment} * \text{Data rate})}$$

Calculations of Offered Load

$$G = \frac{N * P}{C * t_a}$$

G - Offered load

N - Number of nodes

P - Packet length in bits

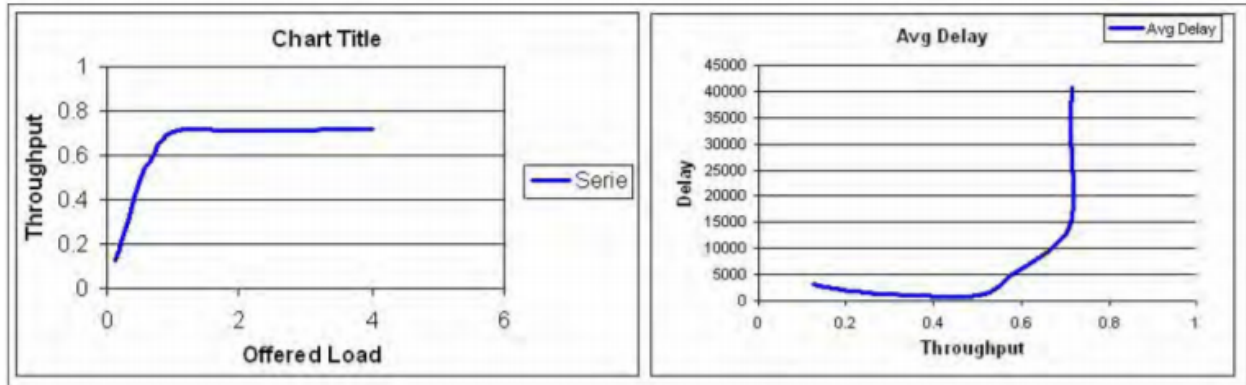
C - Data rate in bits/sec

t_a - Inter packet delay in millisecs.

Model Tabulations

IPD	Tx 1	Tx 2	Tx 3	Tx 4	G - Offered Load	X -Practical Throughput	Avg Delay
16000	7	7	9	5	0.25	0.28	12205.5
8000	11	11	11	13	0.5	0.46	7787.75
4000	18	21	26	19	1	0.84	11265
2000	24	20	21	24	2	0.89	28604.75
1000	20	18	30	20	4	0.88	38216

Model Graph



Experiment - 8

AIM: To simulate and study the link state routing algorithm using simulation.

SOFTWARE REQUIRED: NS-2

THEORY: In **link state routing**, each router shares its knowledge of its neighborhood with every other router in the internet work.

- (i) **Knowledge about Neighborhood:** Instead of sending its entire routing table a router sends info about its **neighborhood** only.
- (ii) **To all Routers:** each router sends this information to every other router on the internet work not just to its neighbor .It does so by a process called **flooding**.
- (iii) **Information sharing when there is a change:** Each router sends out information about the neighbors when there is change.

PROCEDURE:

The Dijkstra algorithm follows four steps to discover what is called the shortest path tree(routing table) for each router: The algorithm begins to build the tree by identifying its roots. The root router's trees the router itself. The algorithm then attaches all nodes that can be reached from the root. The algorithm compares the tree's temporary arcs and identifies the arc with the lowest cumulative cost. This arc and the node to which it connects are now a permanent part of the shortest path tree. The algorithm examines the database and identifies every node that can be reached from its chosen node. These nodes and their arcs are added temporarily to the tree.

The last two steps are repeated until every node in the network has become a permanent part of the tree.

ALGORITHM:

1. Create a simulator object
2. Define different colors for different data flows
3. Open a nam trace file and define finish procedure then close the trace file, and execute nam on trace file.
4. Create n number of nodes using for loop
5. Create duplex links between the nodes
6. Setup UDP Connection between n(0) and n(5)

7. Setup another UDP connection between n(1) and n(5)
8. Apply CBR Traffic over both UDP connections
9. Choose Link state routing protocol to transmit data from sender to receiver.
10. Schedule events and run the program.

PROGRAM:

```
set ns [new Simulator]

set nr [open thro.tr w]

$ns trace-all $nr

set nf [open thro.nam w]

$ns namtrace-all $nf

proc finish { } {

global ns nr nf

$ns flush-trace

close $nf

close $nr

exec nam thro.nam &

exit 0

}

for { set i 0 } { $i < 12 } { incr i 1 } {

set n($i) [$ns node]}

for {set i 0} {$i < 8} {incr i} {

$ns duplex-link $n($i) $n([expr $i+1]) 1Mb 10ms DropTail }

$ns duplex-link $n(0) $n(8) 1Mb 10ms DropTail
```

```
$ns duplex-link $n(1) $n(10) 1Mb 10ms DropTail
$ns duplex-link $n(0) $n(9) 1Mb 10ms DropTail
$ns duplex-link $n(9) $n(11) 1Mb 10ms DropTail
$ns duplex-link $n(10) $n(11) 1Mb 10ms DropTail
$ns duplex-link $n(11) $n(5) 1Mb 10ms DropTail
```

```
set udp0 [new Agent/UDP]
$ns attach-agent $n(0) $udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
set null0 [new Agent/Null]
$ns attach-agent $n(5) $null0
$ns connect $udp0 $null0
set udp1 [new Agent/UDP]
$ns attach-agent $n(1) $udp1
set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 500
$cbr1 set interval_ 0.005
$cbr1 attach-agent $udp1
set null0 [new Agent/Null]
$ns attach-agent $n(5) $null0
$ns connect $udp1 $null0
$ns rtproto LS
$ns rtmodel-at 10.0 down $n(11) $n(5)
```

```
$ns rtmodel-at 15.0 down $n(7) $n(6)
```

```
$ns rtmodel-at 30.0 up $n(11) $n(5)
```

```
$ns rtmodel-at 20.0 up $n(7) $n(6)
```

```
$udp0 set fid_ 1
```

```
$udp1 set fid_ 2
```

```
$ns color 1 Red
```

```
$ns color 2 Green
```

```
$ns at 1.0 "$cbr0 start"
```

```
$ns at 2.0 "$cbr1 start"
```

```
$ns at 45 "finish"
```

```
$ns run
```

OUTPUT:

