Question 1 (4+3+3=10 points) Answer the following questions (in the space provided):

1. **What issues need to be handled if the orientation of a device is changed at the runtime?**

When the orientation of a device is changed at runtime, the following issues need to be handled:Views should be recreated and repositioned to fit the new orientation.Data and state of the app should be preserved to avoid losing user input or progress.Any background tasks or network operations should not be interrupted.

2. **What is the concept of layout and menu inflation?**

Layout inflation is the process of converting an XML layout file into corresponding view objects in memory. This is done using the LayoutInflater class in Android. Menu inflation is a similar process of inflating a menu resource file into menu objects. Both are used to create user interfaces for an Android app.

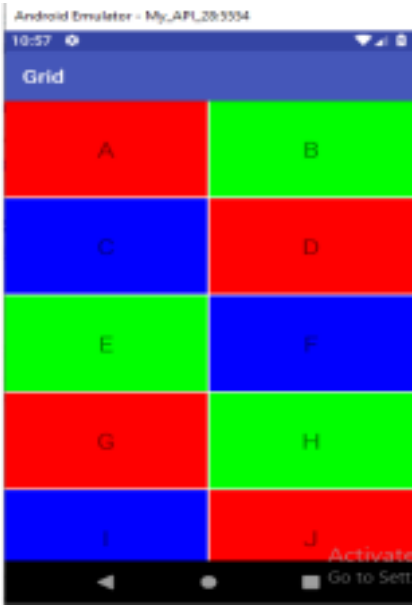
3. **Contrast the memory management approach in Linux and Android.**

The memory management approach in Linux and Android differs in the following ways:In Linux, processes have access to the entire memory of the system, and can allocate and deallocate memory as needed. In Android, each app runs in its own sandboxed environment and has limited access to memory.Linux uses virtual memory to manage memory, which allows processes to access more memory than is physically available. Android also uses virtual memory, but has additional restrictions on memory usage to prevent apps from consuming too much memory and affecting overall system performance.Linux uses a swap space to store pages of memory that are not currently in use. Android also uses a swap space, but has additional mechanisms such as low memory killers to reclaim memory when needed.

Question 2 (20 points)

Red

Blue



Green Red

Suppose we want to use GridView to design a simple two-column layout for displaying alphabets (A-Z), as illustrated in the UI above. Each cell alternates among three colors (red, green and blue).

A basic code skeleton is given. Please fill-in the missing code sections to achieve the required layout. Make use of all the best practices required to handle scrolling of the contents considering re-cycling of views and the View Holder pattern.

Layout Files

activity_main.xml

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
 android:layout_height="match_parent">
 <GridView
 android:id="@+id/grid"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:numColumns="2"
 android:horizontalSpacing="5px"
 android:verticalSpacing="5px"
 />
</LinearLayout>
```

grid_cell.xml

```
<TextView xmlns:android=
 "http://schemas.android.com/apk/res/android"
 android:layout_width="match_parent"
 android:layout_height="1in"
 android:gravity="center"
 android:textSize="24sp"
 >
</TextView>
```

Source Code

```
public class MainActivity extends Activity {
 GridView grid;
 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);


 public class MainActivity extends Activity {
GridView grid;
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
bind();
}
void bind(){
String array[] = {"A","B","C","D","E","F","G","H","I","J","K",
"L","M","N","O","P","Q","R","S","T","U","V","W","X","Y","Z"};
GridAdapter adapter = new GridAdapter(this, array);
grid = findViewById(R.id.grid);
grid.setAdapter(adapter);
}
}

public class GridAdapter extends BaseAdapter {
private final Context context;
private final String[] alphabet;
private final int[] colors = {0xFFFF0000, 0xFF00FF00,
0xFF0000FF}; // {red,green,blue}
private final LayoutInflater inflater;

java
Copy code
public GridAdapter(Context context, String[] alphabet) {
    this.context = context;
    this.alphabet = alphabet;
    inflater = LayoutInflater.from(context);
}

@Override
public int getCount() {
    return alphabet.length;
}

@Override
public Object getItem(int position) {
```

```
    return alphabet[position];
}

@Override
public long getItemId(int position) {
    return position;
}

static class ViewHolder {
    TextView textView;
}

@Override
public View getView(int position, View convertView, ViewGroup
parent) {
    ViewHolder viewHolder;
    if (convertView == null) {
        convertView = inflater.inflate(R.layout.grid_cell, null);
        viewHolder = new ViewHolder();
        viewHolder.textView =
convertView.findViewById(android.R.id.text1);
        convertView.setTag(viewHolder);
    } else {


  bind();
 }
 void bind(){
 String array[] = {"A","B","C","D","E","F","G","H","I","J","K",
"L","M","N","O","P","Q","R","S","T","U","V","W","X","Y","Z"};  //
specify any missing code below in the line
 GridAdapter adapter = ____________________________________________;
grid.setAdapter(adapter);
 }
}
```

```
public GridAdapter(Context context, String[] alphabet) {
    this.context = context;
    this.alphabet = alphabet;
    inflater = LayoutInflater.from(context);
}

@Override
public int getCount() {
    return alphabet.length;
}

@Override
```

```
public Object getItem(int position) {
    return alphabet[position];
}

@Override
public long getItemId(int position) {
    return position;
}

static class ViewHolder {
    TextView textView;
}

@Override
public View getView(int position, View convertView, ViewGroup parent) {
    ViewHolder viewHolder;
    if (convertView == null) {
        convertView = inflater.inflate(R.layout.grid_cell, null);
        viewHolder = new ViewHolder();
        viewHolder.textView =
convertView.findViewById(android.R.id.text1);
        convertView.setTag(viewHolder);
    } else {
        viewHolder = (ViewHolder) convertView.getTag();
    }

    viewHolder.textView.setText(alphabet[position]);

    int color = colors[position % colors.length];
    viewHolder.textView.setBackgroundColor(color);

    return convertView;
}
```