1. What's your favorite tool or library for Android? Why is it so useful?

My favorite Android tool is Android Lint which scans Android project sources for potential bugs such as: -

- Missing translations (and unused translations)
- Layout performance problems (all the issues the old layoutopt tool used to find, and more)
- Unused resources
- Inconsistent array sizes (when arrays are defined in multiple configurations)
- Accessibility and internationalization problems (hard coded strings, missing contentDescription, etc.)
- Icon problems (like missing densities, duplicate icons, wrong sizes, etc.)
- Usability problems (like not specifying an input type on a text field)
- Manifest errors

2. You want to open a map app from an app that you're building. The address, city, state, and ZIP code are provided by the user. What steps are involved in sending that data to a map app?

```
Uri mapUri = Uri.parse("geo:0,0?q=address + city + state + ZIP code");
Intent mapintent = new Intent(Intent.ACTION_VIEW, mapUri );
mapintent.setPackage("com.google.android.apps.maps");
PackageManager packageManager = getPackageManager();
List<ResolveInfo> activities = packageManager.queryIntentActivities(mapintent, 0);
boolean isIntentSafe = activities.size() > 0;
 if (isIntentSafe) {
startActivity(mapintent);
}
```

3. String compression using the counts of repeated characters. For example, the string aabcccccaaa would become a2b1c5a3.

```
import java.util.Scanner;

public class test {
	public static void main(String args[]) {
		Scanner scan=new Scanner(System.in);
		String longstring=scan.nextLine();
		String compresstr=compress(longstring);
		if(compresstr.length()> longstring.length()) {
			System.out.println(longstring);
		}
		else {
			System.out.println(compresstr);
		}
	}


 public static String compress(String input){
```

```
  String compresstring="";
for(int i=0;i<input.length();i++){
        char s=input.charAt(i);
        int count=0;
for(int j=i;j<input.length();j++){
        char s1=input.charAt(j);
        if(Character.toString(s1).equals(Character.toString(s))){
                count++;
                if(j == input.length() -1){
                  compresstring=compresstring+ Character.toString(s)+count + "";
                      i=j;
                }
        }
 else{
          i=j-1;
          compresstring=compresstring + Character.toString(s)+ count +"";
          break;
}
}

}
 return compresstring;

}
}
```

4. List and explain the differences between four different options you have for saving data while making an Android app.

1. SharedPreferences
2. Internal storage with files
3. External storage with sd card
4. SQL Databases

SQL Databases: - Saving data to a database is ideal for repeating or structured data, such as contact information. To implement Database we need to follow these steps:-

1. First, we need to define the schema of the database. So we create an inner class which implements BaseColumns Interface. As a result, it can inherit a primary key field called _ID. In this class, we declare table name and column names.
2. Next, we create Database Helper class which inherits from SQLiteOpenHelper class. In this class, we write the commands to create the table with each column specifying the type of data it can store. We also declare the Database version variable, which increments, if we change the schema of Database anytime.
3. To continue, we create a class that inherits from Content providers. Content providers can help an application manage access to the database stored by itself, stored by other apps, and provides a way to share data with other apps. We implement insert(), query(), delete() methods for the database in this class. Content Providers provides an abstraction to the Database. This abstraction

allows us to make modifications to our application database without affecting other existing applications that rely on our database.

4. To conclude we use getContentResolver() in Activity to access the content provider methods.

5. What are your thoughts about Fragments? Do you like or hate them? Why?

Fragments are a piece of an application's user interface or behavior that can be placed in an Activity. Interaction with fragments is done through Activity.getFragmentManager(). A Fragment is closely tied to the Activity it is in, and can not be used apart from one. Fragment defines its own lifecycle, but that lifecycle is dependent on its activity.

I like Fragments, as it is very useful for the mobiles and tablets with big screens. When single Activity layout cannot fill the UI screen completely, we can combine multiple fragments in a single Activity to build a multi-pane UI. We can also reuse fragment in multiple Activities.

6. If you were to start your Android position today, what would be your goals a year from now?

A year from now, I would like to see myself as a person who has learned and is more proficient than what I am today. To achieve that, I will put my sincere effort in all my responsibilities such as unit-test coding, bug fixing, designing and building advanced applications, etc. I also like to add value to this position by having same mission for building the next generation mobile applications in the sphere of global capacity delivery, logistics, compliance and inventory management and move into roles with greater responsibilities.