a)

(i) Application Framework: It refers to a set of software libraries, tools, and components that provide a foundation for developing applications. In mobile app development, an application framework often includes APIs for interacting with the device's hardware, managing user interfaces, handling networking, and more.

(ii) Activity: In the context of Android development, an activity represents a single screen with a user interface. It is a fundamental building block of an Android application, where each activity typically corresponds to a different UI screen that the user can interact with.

(iii) Window: In mobile application development, a window generally refers to a container that holds the user interface elements, such as views, buttons, and text fields. Windows are often managed by the operating system and are used to display the various screens of an application.

(iv) Widget: A widget is a small, reusable UI component that can be embedded within an activity or placed on the home screen of a mobile device. Examples of widgets include buttons, text fields, checkboxes, and sliders. Widgets enhance the user experience by providing quick access to frequently used functionality.

(v) User Experience: User experience (UX) encompasses all aspects of the end-user's interaction with a mobile application. It includes factors such as ease of use, intuitiveness, accessibility, and overall satisfaction with the app. Designing for a positive user experience involves understanding user behaviors, preferences, and needs.

b)

Two advantages of Android development environment over the iOS environment:

Open Ecosystem: Android offers a more open ecosystem compared to iOS, allowing developers more flexibility in customizing and distributing their applications. Developers can easily publish apps on the Google Play Store without stringent approval processes, and they have greater control over device functionalities such as file management and system customization.

Fragmentation Management: While fragmentation can be seen as a challenge in Android development due to the diversity of devices and OS versions, it also offers an advantage in terms of market reach. Android developers can target a wider audience by supporting a variety of screen sizes, resolutions, and hardware configurations, whereas iOS development is more streamlined with fewer device variations.

c)

Three types of mobile applications:

Native Applications: These are developed for a specific mobile platform (e.g., iOS, Android) using platform-specific programming languages (Swift or Objective-C for iOS, Java or Kotlin for Android). Native apps offer high performance and access to platform-specific features but require separate development efforts for each platform.

Web Applications: Web apps are accessed through a mobile device's web browser and are developed using web technologies such as HTML, CSS, and JavaScript. They offer cross-platform compatibility but may have limitations in accessing device hardware and native functionalities.

Hybrid Applications: Hybrid apps combine elements of both native and web applications. They are built using web technologies but are wrapped in a native container, allowing them to access device features like camera and GPS. Hybrid apps offer faster development time and easier maintenance compared to native apps but may suffer from performance issues.

d)

(i) Relative Layout vs. Absolute Layout:

Relative Layout: It allows positioning UI elements relative to each other or relative to the parent layout. This provides flexibility in adapting to different screen sizes and orientations.

Absolute Layout: It involves specifying the exact position (in terms of pixels) of UI elements on the screen. While it offers precise control over element placement, it may not scale well across different devices with varying screen sizes and resolutions.

(ii) Activities vs. Fragments:

Activities: Activities represent individual screens with a user interface in an Android application. They are typically used to present a single focused interaction to the user, such as viewing a list of items or editing a form.

Fragments: Fragments are reusable UI components within an activity that can be combined to create a multi-pane UI. They are useful for creating flexible layouts that can adapt to different screen sizes and orientations. Fragments can be dynamically added, removed, or replaced within an activity to provide a modular and flexible user interface.

e)

Three types of testing required before a mobile application is deployed:

Functional Testing: This involves testing the application's features and functionality to ensure that it behaves as expected according to the requirements. It includes testing user interactions, input validation, data processing, and error handling.

Performance Testing: Performance testing assesses the application's responsiveness, stability, and resource usage under various conditions such as different network speeds, device configurations, and user loads. It aims to identify and address performance bottlenecks, memory leaks, and other issues that could degrade the user experience.

Compatibility Testing: Compatibility testing verifies that the application works correctly across different devices, operating system versions, screen sizes, and orientations. It ensures that the app's layout, functionality, and performance remain consistent across various platforms and configurations.

f)

Two features of the Android mobile development environment that have made it popular:

Open-Source Platform: Android is built on an open-source Linux kernel, which encourages collaboration and innovation among developers worldwide. The availability of the Android Open Source Project (AOSP) allows developers to customize the platform according to their needs, leading to a diverse range of devices and applications.

Large User Base: Android has a significantly larger user base compared to other mobile platforms, making it an attractive market for developers. The vast user base presents greater opportunities for reaching a wider audience and monetizing applications through various distribution channels such as the Google Play Store.

QUESTION TWO [20 MARKS]

a) The six steps of the mobile application development process are:

Planning and Analysis: This phase involves defining the scope, objectives, and requirements of the mobile application. It includes conducting market research, identifying target users, defining features and functionalities, and creating wireframes or prototypes.

Design: In this phase, the visual and user interface (UI) design of the application is created based on the requirements gathered during the planning phase. Designers focus on creating an intuitive and visually appealing user experience (UX) by considering factors such as navigation, layout, color schemes, and branding.

Development: The development phase involves writing code to implement the features and functionalities of the mobile application. Developers use programming languages such as Java, Kotlin (for Android), Swift (for iOS), or HTML/CSS/JavaScript (for hybrid apps) to build the app's logic, user interface, and backend infrastructure.

Testing: During the testing phase, the mobile application is rigorously tested to identify and fix any bugs, errors, or usability issues. Various types of testing are performed, including functional testing, performance testing, compatibility testing, and user acceptance testing (UAT), to ensure the quality and reliability of the application.

Deployment: Once the mobile application has been thoroughly tested and approved, it is ready for deployment to the target platform(s). Deployment involves packaging the application, submitting it to the respective app stores (e.g., Google Play Store, Apple App Store), and complying with the store's guidelines and requirements.

Maintenance and Updates: After the application is deployed, ongoing maintenance and updates are necessary to address user feedback, fix bugs, add new features, and ensure compatibility with evolving technologies and platforms. Maintenance activities may include monitoring app performance, analyzing user metrics, and releasing periodic updates.

b)

(i) Considerations for selecting a mobile application development platform:

Target Audience and Market: The choice of development platform should align with the target audience's preferences and the market demand. Factors such as demographics, device usage trends, and geographic location can influence the platform selection.

Development Resources

a)

Five Android layout types and their applications:

LinearLayout: LinearLayout arranges its children elements linearly either horizontally or vertically. It is suitable for creating simple user interfaces with elements aligned in a single direction, such as forms or lists.

RelativeLayout: RelativeLayout enables positioning of child views relative to each other or the parent layout. It is useful for creating complex user interfaces where the positioning of elements needs to be flexible and adaptable to different screen sizes.

ConstraintLayout: ConstraintLayout allows creating complex layouts by defining constraints between views. It is highly flexible and efficient for building responsive user interfaces that adapt well to various screen sizes and orientations.

FrameLayout: FrameLayout is designed to display a single item at a time. It is commonly used for displaying fragments or switching between different views or screens in an application.

GridLayout: GridLayout arranges its children elements in a grid-like fashion, with rows and columns. It is suitable for creating tabular layouts or grids of items, such as photo galleries or data tables.

b)

Five methods used to describe the fragment lifecycle in Android application development:

onCreate(): This method is called when the fragment is first created. It is typically used for initializing essential components and preparing the fragment for user interaction.

onCreateView(): This method is responsible for creating the fragment's view hierarchy, usually by inflating a layout XML file. It is where the fragment's UI elements are initialized and configured.

onStart(): This method is invoked when the fragment becomes visible to the user. It is a good place to perform tasks like refreshing data or starting animations.

onResume(): This method is called when the fragment is visible and actively interacting with the user. It is where the fragment should resume any operations that were paused or stopped.

onPause(): This method is triggered when the fragment is about to lose focus or become partially obscured. It is used to pause ongoing tasks or save the fragment's state to ensure a smooth transition between states.

QUESTION FIVE [20 MARKS]

a)

Four parts of a GSM network and their brief descriptions:

Mobile Station (MS): The mobile station refers to the physical device used by the end-user, typically a mobile phone or a GSM modem. It consists of the mobile equipment (ME), which includes the handset or device, and the Subscriber Identity Module (SIM) card.

Base Station Subsystem (BSS): The BSS comprises two main components: Base Transceiver Station (BTS) and Base Station Controller (BSC). The BTS is responsible for transmitting and receiving radio signals to and from the mobile stations within its coverage area. The BSC manages and controls multiple BTSs, handling functions such as handovers and frequency allocation.

Network Switching Subsystem (NSS): The NSS consists of several components, including the Mobile Switching Center (MSC), Visitor Location Register (VLR), and Home Location Register (HLR). The MSC is the central component responsible for call switching, mobility management, and other core network functions. The VLR stores temporary subscriber information for users roaming in its coverage area, while the HLR contains permanent subscriber data and manages subscriber authentication and call routing.

Operations Support System (OSS): The OSS includes various components and systems for network management, monitoring, and administration. It encompasses functionalities such as performance monitoring, fault management, configuration management, and billing systems.

b)

Categories of Android application components:

Activities: Activities represent individual screens with a user interface in an Android application. They serve as the entry point for user interaction and typically correspond to a single focused task or user interface.

Services: Services are background processes that run independently of the user interface and can perform long-running operations or tasks in the background, such as playing music, downloading files, or handling network requests.

Broadcast Receivers: Broadcast Receivers are components that listen for and respond to system-wide broadcast messages or intents. They enable communication between different applications or system components and allow reacting to system events or notifications.

Content Providers: Content Providers manage and share structured data with other applications. They encapsulate data storage, retrieval, and access mechanisms, providing a consistent interface for accessing shared data such as contacts, media files, or calendar events.

c)

The role of a mobile emulator in application development is to simulate the behavior and characteristics of a real mobile device on a computer. Emulators allow developers to test their applications in a controlled environment without the need for physical devices, which can be costly and limited in availability. They provide features such as simulating different screen sizes, resolutions, hardware configurations, and network conditions, allowing developers to ensure that their applications function correctly and perform well across various device specifications. Emulators also support debugging and testing features, enabling developers to identify and fix issues efficiently during the development process. Overall, mobile emulators play a crucial role in accelerating development, improving application quality, and enhancing the user experience.

a. Android applications include components that may be classified into four categories:

Activities: Activities represent the UI and behavior of a single screen in an application. They are responsible for interacting with the user and handling user input, such as button clicks or text input.

Services: Services are background components that perform long-running operations or tasks without a UI. They run independently of the user interface and are commonly used for tasks like playing music, downloading files, or syncing data in the background.

Broadcast Receivers: Broadcast Receivers listen for and respond to system-wide broadcast messages or intents. They enable communication between different components within an application or between different applications and allow reacting to system events or notifications.

Content Providers: Content Providers manage and share structured data with other applications. They encapsulate data storage, retrieval, and access mechanisms, providing a consistent interface for accessing shared data such as contacts, media files, or calendar events.

b. Files for a simple tax calculator:

i. MainActivity.java:

java

Copy code

import android.os.Bundle;

import android.widget.EditText;

import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

EditText incomeInput;

TextView taxOutput;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

incomeInput = findViewById(R.id.incomeInput);

taxOutput = findViewById(R.id.taxOutput);

}

// Method to calculate tax

public double calculateTax(double income) {

// Tax calculation logic

return income \* 0.2; // Example tax rate

}

}

ii. Main\_Activity.xml:

xml

Copy code

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

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

android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

<EditText

android:id="@+id/incomeInput"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:hint="Enter income"

android:layout\_centerHorizontal="true"

android:layout\_marginTop="50dp"/>

<Button

android:id="@+id/calculateButton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Calculate"

android:layout\_below="@id/incomeInput"

android:layout\_centerHorizontal="true"

android:layout\_marginTop="20dp"/>

<TextView

android:id="@+id/taxOutput"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Tax: "

android:layout\_below="@id/calculateButton"

android:layout\_centerHorizontal="true"

android:layout\_marginTop="20dp"/>

</RelativeLayout>

iii. Manifest.xml:

xml

Copy code

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

package="com.example.taxcalculator">

<application

android:allowBackup="true"

android:icon="@mipmap/ic\_launcher"

android:label="@string/app\_name"

android:roundIcon="@mipmap/ic\_launcher\_round"

android:supportsRtl="true"

android:theme="@style/AppTheme">

<activity android:name=".MainActivity">

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

</activity>

</application>

</manifest>

iv. Incometax.java:

java

Copy code

public class Incometax {

public static void main(String[] args) {

double income = 50000; // Example income

double tax = calculateTax(income);

System.out.println("Tax: " + tax);

}

public static double calculateTax(double income) {

// Tax calculation logic

return income \* 0.2; // Example tax rate

}

}

c. Android operating systems for mobile devices have been increasing their market share worldwide due to their diverse range of devices, customization options, and affordability. One reason for this trend is the wide variety of Android-powered devices available across different price ranges, catering to a diverse global audience. Unlike iOS, which is limited to Apple devices, Android is available on smartphones and tablets from various manufacturers, offering consumers a broader choice of hardware options with varying features, specifications, and price points. This versatility appeals to a wide range of users, including those in emerging markets or budget-conscious consumers, contributing to Android's growing market share.

d. Default parameters refer to the values assigned to function parameters if no argument is provided when calling the function. Overloading, on the other hand, involves defining multiple functions with the same name but different parameter lists. The relationship between default parameters and overloading lies in their ability to provide flexibility and convenience in function definitions. Default parameters allow a function to have optional arguments, simplifying function calls and reducing the need for overloaded functions to handle different parameter combinations explicitly. By specifying default values for certain parameters, a single function definition can accommodate various use cases, reducing code duplication and improving readability. Overloading, meanwhile, enables defining multiple versions of a function with different parameter types or numbers, providing versatility in function usage and allowing developers to choose the appropriate version based on their requirements. Together, default parameters and overloading contribute to code clarity, flexibility, and reusability in software development.

e. Four parts of a GSM network:

Mobile Station (MS): The mobile station refers to the physical device used by the end-user, typically a mobile phone or GSM modem. It consists of the mobile equipment (ME), which includes the handset or device, and the Subscriber Identity Module (SIM) card.

Base Station Subsystem (BSS): The BSS comprises two main components: Base Transceiver Station (BTS) and Base Station Controller (BSC). The BTS is responsible for transmitting and receiving radio signals to and from the mobile stations within its coverage area. The BSC manages and controls multiple BTSs, handling functions such as handovers and frequency allocation.

Network Switching Subsystem (NSS): The NSS consists of several components, including the Mobile Switching Center (MSC), Visitor Location Register (VLR), and Home Location Register (HLR). The MSC is the central component responsible for call switching, mobility management, and other core network functions. The VLR stores temporary subscriber information for users roaming in its coverage area, while the HLR contains permanent subscriber data and manages subscriber authentication and call routing.

Operations Support System (OSS): The OSS includes various components and systems for network management, monitoring, and administration. It encompasses functionalities such as performance monitoring, fault management, configuration management, and billing systems.

f. Three limitations associated with mobile devices:

Limited Processing Power: Mobile devices, especially budget or mid-range smartphones, often have limited processing power compared to desktop computers or laptops. This limitation can impact the performance of resource-intensive applications, leading to slow response times, laggy user interfaces, or crashes.

Limited Storage Capacity: Mobile devices typically have limited internal storage capacity, which can restrict the amount of data, apps, and media files that users can store on their devices. This limitation can necessitate frequent data management, including deleting files or uninstalling apps to free up space.

Limited Battery Life: Mobile devices rely

Diagram of the four layers of the Android software architecture:

markdown

Copy code

--------------------------

| Application Layer |

--------------------------

| Application Framework|

--------------------------

| Libraries |

--------------------------

| Linux Kernel |

--------------------------

b. Information hiding is a principle in software engineering where details of a module or component are hidden from the rest of the system, revealing only the essential interfaces or functionalities. It is implemented using encapsulation, which involves bundling data and methods that operate on the data within a single unit, such as a class in object-oriented programming. By encapsulating data and methods, access to the internal workings of the module is restricted, and only designated interfaces are exposed to the external components. This helps in reducing complexity, managing dependencies, and facilitating modular design and maintenance.

c.

i) Relative Layout:

xml

Copy code

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

android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

<Button

android:id="@+id/button1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Button 1"

android:layout\_alignParentTop="true"

android:layout\_alignParentStart="true"/>

<Button

android:id="@+id/button2"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Button 2"

android:layout\_below="@id/button1"

android:layout\_alignParentEnd="true"/>

</RelativeLayout>

ii) Frame Layout:

xml

Copy code

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

android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

<ImageView

android:id="@+id/imageView"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:src="@drawable/image"/>

<TextView

android:id="@+id/textView"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Hello, Frame Layout!"

android:textSize="24sp"

android:layout\_gravity="center"/>

</FrameLayout>

d.

Functions of a manifest XML file in mobile programming:

Declaring Components: The manifest file declares all components of the application, including activities, services, broadcast receivers, and content providers. This declaration is necessary for the Android system to identify and interact with the components correctly.

Defining Permissions: The manifest file specifies the permissions required by the application to access sensitive system resources or perform restricted operations. This ensures that the user is informed about the capabilities of the application and can grant or deny permissions accordingly.

Setting Hardware and Software Features: The manifest file specifies the hardware and software features required by the application, such as camera, GPS, accelerometer, or minimum Android version. This information helps the system determine compatibility with the device and filter the application accordingly in the app store.

Configuring Application Metadata: The manifest file includes metadata about the application, such as its name, version, icon, label, and theme. This metadata is used for displaying the application in the app launcher, app store, or system settings, and for identifying and managing the application on the device.