TASK : 1

1)Why Kotlin is preferred over other languages for android application development ? state five reason.

ANS: Developed by JetBrains, Kotlin is an open-source, statically-typed programming language for JavaScript and Java Virtual Machine (JVM). It's used by several big brands like Uber, Pinterest, Trello, and Twitter.

Kotlin attempts to solve many of the challenges developers face with older languages like Java. Java is verbose, throws frequent errors due to Null Pointer Exceptions and nullability, and poses the risk of memory leaks due to the way inner classes are implemented. Kotlin is both simpler and more powerful than Java. It also has some standout features that set it apart from modern frameworks like Flutter, such as its better performance and scalability.

Faster development times and reduced costs thanks to Kotlin being a much more concise language than Java, meaning you can write less code and achieve the same results. Many features that are specifically designed for Android development, such as built-in support for null safety and Android Extensions..

State Five Reasons kotlin is preferred over other languages for Android Application Development:

They are :

1)Easy to setup

2) Completely Interoperable with Java

3)Open Source

4) Reduces Boilerplate Code

5)No More Null Pointer Exceptions

1. Easy to Setup:

Now Kotlin comes up with easy setup feature. Before, developers were forced to use a plugin to add Kotlin into their Android projects. But now Android studio 3.0 comes up with an in-built feature of “Support Kotlin Development”, which has become a whole lot easier. You just need to check the ‘Include Kotlin Support’ box while configuring your new project in Android studio and you are done then.

2. Completely Interoperable with Java:

One of the greatest strengths of Kotlin is Interoperability. It’s completely and 100% interoperable with Java. The beautiful essence of this feature is that it allows developers to use Java and Kotlin code side by side in the same project without disturbing any of the factors.

This means if you are planning to integrate Kotlin in your existing project, you are just required to add its tools and frameworks without any additional changes.

3. Open Source:

Kotlin is an open-source programming language which reduces the time with converting existing Java code in a single-click tool. Being an open-source language, Kotlin supports core product development.

4. Reduces Boilerplate Code:

Code less. Debug less.

Nobody likes to write lengthy lines of code. Right? This is another feature Kotlin providing to developers. Boilerplate code is code that you need to write again and again with no alterations. Thus, Kotlin reduces boilerplate code. With the help of various handy tricks and methods, Kotlin reduces the extra lines of codes and create a relatively smaller file size. The reduced

amount of code is almost 20%. The fewer lines of code will enhance the speed of your app development lifecycle.

5. No More NullPointerExceptions:

Also known as ‘the billion dollar mistake’, NullPointerExceptions (NPE) has been one of the most common causes of app crash. It removes the null reference from the code. It’s a quite time-consuming and tough task for developers to fix the NullPointerExceptions and protect your code using null checks. But by utilizing Kotlin, NPE is a thing of past as null safety comes baked into the language’s type system. And this lets you catch the NPEs during the compile-time instead of the app crash at runtime.

Kotlin allows you to write less and hassle-free code than Java. It’s one of the most significant advantages of using Kotlin for Android app development. This feature is really valuable for the developers because while handling the Java programming language NullPointerException takes place lots of times and certain elements used to test the software quality will decrease the development speed and efficiency.

These are the five reasons kotlin is preferred the over other languages than android applicaton development.

TASK: 2

2)Write a kotlin program to generate the following output.

Note: The name can be different.

ANS: KOTLIN PROGRAM:

```
fun main()
{
        println("Happy Birthday,Nandini!")
        println("  ,,,,,  ")
        println("  |||||  ")
       println(" ========= ")
       println("@@@@@@@@@@@")
        println("{~@~@~@~@~}")
      println("@@@@@@@@@@@")
}
```

OUTPUT:

```
Happy Birthday,Nandini!
  ,,,,,
  |||||
 =========
@@@@@@@@@@
{~@~@~@~@~}
@@@@@@@@@@
```

TASK: 3

3)How to declare a variable in kotlin ? Elaborate difference between var and val by taking suitable examples.

ANS: Kotlin Variable:

Variable refers to a memory location. It is used to store data. The data of variable can be changed and reused depending on condition or on information passed to the program.

Syntax:

var variablename=value

val variablename=value

Example:

var name= "nandu"

val birthyear=2000

println(name)

println(birthyear)

Variable Declaration In Kotlin:

Kotlin variable is declared using keyword **var** and **val**.

1. var language ="Java"
2. val salary = 30000

Example:

We can also explicitly specify the type of variable while declaring it.

1. var language: String ="Java"

2. val salary: Int = 30000

   It is not necessary to initialize variable at the time of its declaration. Variable can be initialized later on when the program is executed.

1. var language: String
2. ... ...
3. language = "Java"
4. val salary: Int
5. ... ...
6. salary = 30000

Rules for defining variables In Kotlin:

A variable can have a short name (like x and y) or more descriptive names (age, sum, totalVolume).

The general rule for Kotlin variables are:

- Names can contain letters, digits, underscores, and dollar signs
- Names should start with a letter
- Names can also begin with $ and _ (but we will not use it in this tutorial)
- Names are case sensitive ("myVar" and "myvar" are different variables)
- Names should start with a lowercase letter and it cannot contain whitespace
- Reserved words (like Kotlin keywords, such as `var` or `String`) cannot be used as names.

Difference between var and val in Kotlin:

var and **val** are both used to declare variables in Kotlin language. However, there are some key differences between them:

VAR(Variable)

It is a general variable. The value of a variable that is declared using var can be changed anytime throughout the program. var is also called **mutable** and **non-final variable**, as there value can be changed anytime.

Example 1:

```
fun main()

 {

   var marks = 10

   println("Previous marks is " + marks)

   marks = 30

   println("New marks " + marks)

 }
```

Output :

```
Previous marks is 10

New marks 30
```

Example 2:

```
fun main()

{

  val per = 75
  println ("Marks" + per)
  per = 85
  println ("new per " + per)
```

```
}
```

VAL(Value)

The object stored using val cannot be changed, it cannot be reassigned, it is just like the **final** keyword in java. val is **immutable**. Once assigned the val becomes read-only, however, the properties of a val object could be changed, but the object itself is read-only.

Example 1:

```
fun main()

{

    val marks = 10

    println("Previous marks is " + marks)

    marks = 30

    println("new marks " + marks)

}
```

Output:

```
Val cannot be reassigned
```

Example 2:

```
// Changing values of val object

fun main()

{

    val book = Book("Java", 1000)

    println(book)

    book.name = "Kotlin"

    println(book)

}

data class Book(var name : String = "",

                var price : Int = 0)
```

output:

```
Book(name=Java, price=1000)
Book(name=Kotlin, price=1000)
```

Example in using var and val in kotlin variables:

```
val sName = "Kartexa Developer";
var varName = "World";

fun main() {
```

```
   println("Example of val--->"+sName);
   println("Example of Var--->"+varName);

   // Variable declared by var is mutable
   varName = "new value";
   println("New value of the variable declared using Var: "
+varName);
}
```

OUTPUT:

It will generate the following output −

Example of val--->Kartexa Developer

Example of Var--->World

New value of the variable declared using Var: new value