Name - Akshay Patwal
Father's Name — Mohan Singh Patwal

Subject — Scala
Subject Code — TCS-532

Sem - 5th
Course - B.Tech (DS & AI)
Type - Regular

Akshay

Q1 (a)

Scala supports nested functions. They are defined as function inside an another function.

Nested functions are not supported by other languages like C++, Java and many more. In Scala we can however, define a function inside another function, called nested function or local function.

Syntax :

```
def functionName1 ( Parameter 1, Parameter 2 )=
{
    //Code
    def functionName2 ( ) = {
     //code
    }
}
```

Nested functions are basically of 2 types in Scala:

1. Single Nested function.
2. Multiple Nested function.

Nested functions make it easier to detect code and increase modularity.

example :-

```
object Myclass {
    def factorial (n:Int): Int = {
```

Akshay

```
def fact ( n: Int, accumulator : Int): Int = {
    if ( n <= 1)
       accumulator
    else
       fact ( n-1, n* accumulator )
  }
  fact ( n, 1 )
}
def main (args: Array[String]){
    println ("fact of 10 is :" + factorial (10));
}
```

Scala supports higher-order functions :

A higher order function takes other functions as parameter or return a function as a result.

This is possible because functions are first-class value in Scala. It means that functions can be passed as argument to other functions and also a function can return other function.

eg → map function is classic example of higher order function.

```
val list = List (1, 2, 3)
```

Akshay

Q2 (b)

i) Object in scala can be mutable or immutable.

Mutable :- Mutable type are those which allow us to change values after declaration. To make an object mutable in scala we declare it using 'var' keyword.

Syntax : var variable_name : datatype = "value";

example : var name : string = "HelloWorld";

Immutable :- Immutable type are those which don't allow us to change values after declaration. for declaring immutable objects in scala we use 'val' keyword.

Syntax : val variable_name : datatype = "value";

example : val name : string = "HelloWorld";

Now, we can't change value of name from 'HelloWorld' to other.

ii) Closure function & Anonymous function.

Closure function :- A closure can be defined as a function whose return value depend on value of one or more variables defined outside function.

Akshay

eg :-

```
val b = 100
def Example (50 : double) = (50 * b) / 100
```

Anonymous function :- Anonymous function is also known as function literal. A function which doesn't contain a name, is known as anonymous function.

An anonymous function provides a light weight function defination. It is useful when we want to create Inline function.

Syntax :

```
(z : Int, y : Int) => z * y
```

(or)

```
(_ : Int) * (_ : Int)
```

eg →

```
object Main {
    def main (args : Array [String]) {
        var myFC1 = (str1 : String, str2 : String) => str1 + str2
        var myFC2 = (_ : String) + (_ : String)
        println (myFC1 ("Hello", "World"))
        println (myFC1 ("Hello", "World"))
    }
}
```

Anonymous function

⇒ Both FC1 & FC2 print "HelloWorld".

Akshay