**Task 1**

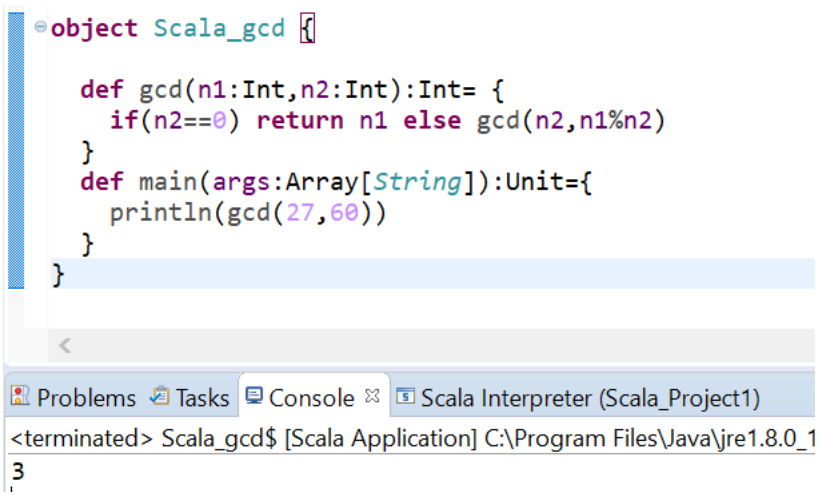
**Create a Scala application to find the GCD of two numbers**

**Code-**

```
object Scala_gcd {

  def gcd(n1:Int,n2:Int):Int= {
    if(n2==0) return n1 else gcd(n2,n1%n2)
  }
  def main(args:Array[String]):Unit={
    println(gcd(27,60))
  }
}
```

**Output-**



```
object Scala_gcd {

  def gcd(n1:Int,n2:Int):Int= {
    if(n2==0) return n1 else gcd(n2,n1%n2)
  }
  def main(args:Array[String]):Unit={
    println(gcd(27,60))
  }
}
```

```
<terminated> Scala_gcd$ [Scala Application] C:\Program Files\Java\jre1.8.0_1
3
```

**Task 2**

**Fibonacci series (starting from 1) written in order without any spaces in between, thus producing a sequence of digits.**
**Write a Scala application to find the Nth digit in the sequence.**

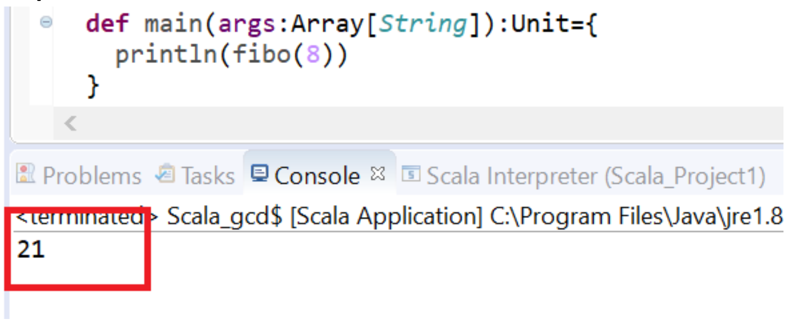
➢ **Write the function using standard for loop**
**Code-**

```
object Scala_fibo {

  def fibo(n:Int):Int={
    var first:Int=1
    var second:Int=1
    var count:Int=1
    var sum:Int=0
    while(count<n)
    {
      sum=first+second;
      first=second;
      second=sum;
      count=count+1;
    }
    return first
  }

  def main(args:Array[String]):Unit={
    println(fibo(8))
  }
}
```

**Output-**



```
  def main(args:Array[String]):Unit={
    println(fibo(8))
  }
```

Problems Tasks Console Scala Interpreter (Scala_Project1)

```
<terminated> Scala_gcd$ [Scala Application] C:\Program Files\Java\jre1.8
21
```

➢ **Write the function using recursion**

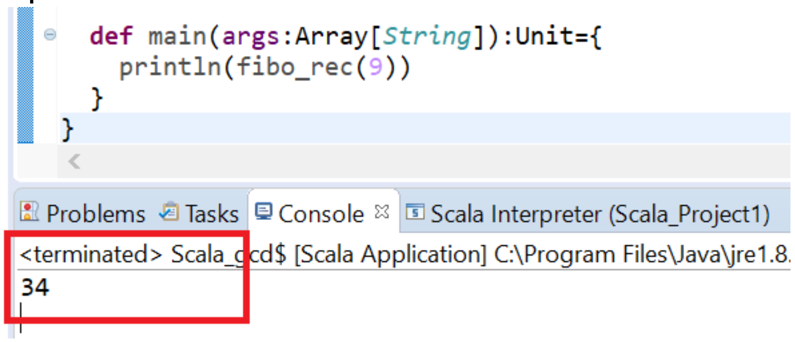
**Code-**

```
object Scala_gcd {

  def fibo_rec(n:Int):Int={
    if(n==0 || n==1)
      return n
    else fibo_rec(n-1)+fibo_rec(n-2);
  }

  def main(args:Array[String]):Unit={
    println(fibo_rec(9))
  }
}
```

**Output-**



```
  def main(args:Array[String]):Unit={
    println(fibo_rec(9))
  }
}
```

Problems Tasks Console Scala Interpreter (Scala_Project1)

```
<terminated> Scala_gcd$ [Scala Application] C:\Program Files\Java\jre1.8.
34
```

**Task 3**

**Find square root of number using Babylonian method.**

**1. Start with an arbitrary positive start value x (the closer to the root, the better).**

**2.Initialize y = 1.**

**3. Do following until desired approximation is achieved.**

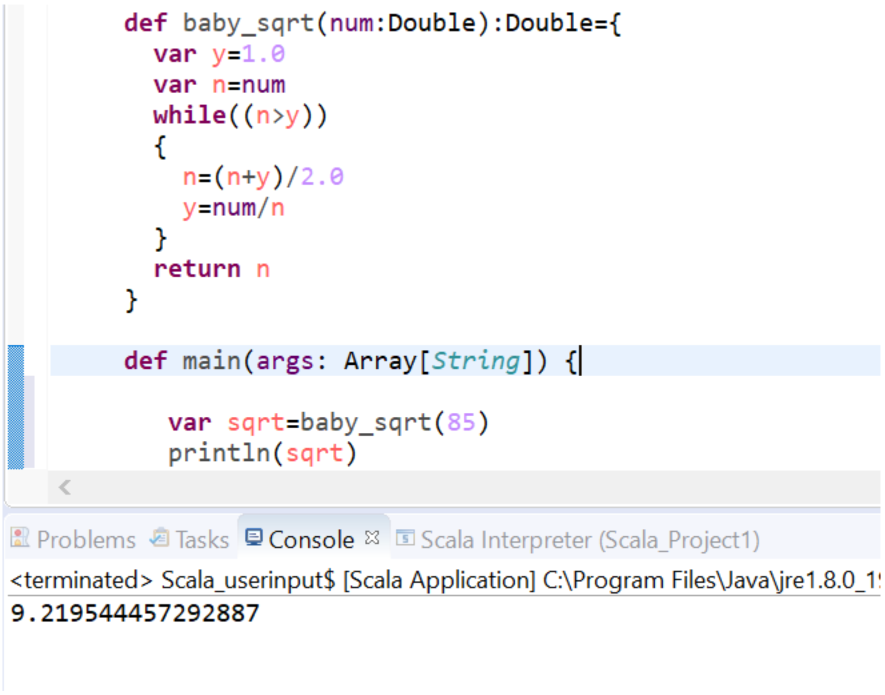
**a) Get the next approximation for root using average of x and y**

**b) Set y = n/x**

**Code-**

```
def baby_sqrt(num:Double):Double={
        var y=1.0
        var n=num
        while((n>y))
        {
          n=(n+y)/2.0
          y=num/n
        }
        return n
      }
```

**Output-**



```
def baby_sqrt(num:Double):Double={
  var y=1.0
  var n=num
  while((n>y))
  {
    n=(n+y)/2.0
    y=num/n
  }
  return n
}

def main(args: Array[String]) {

   var sqrt=baby_sqrt(85)
   println(sqrt)
```

```
<terminated> Scala_userinput$ [Scala Application] C:\Program Files\Java\jre1.8.0_1
9.219544457292887
```