

**Institute of Engineering and Technology (IET)**

**JK Lakshmipat University**

**Kotlin -Assignment**  
*Set 1*

**Submitted BY**

Siddhi Nyati (2022btech101)

**FACULTY GUIDE**

Dr. Pranab Roy

**March 2025**

**Question 1**

**Code:**fun main() {  
 *print*("Enter Gross Income (INR): ")  
 val grossIncome = *readLine*()?.*toDoubleOrNull*() ?: return  
  
 val netIncome = grossIncome - 75000  
 val tax = *calculateIncomeTax*(netIncome)  
 val eduCess = tax \* 0.02  
 val higherEduCess = tax \* 0.01  
 val totalTax = tax + eduCess + higherEduCess  
 val finalIncome = netIncome - totalTax  
  
 *println*("Total Tax Deduction: INR %.2f".*format*(totalTax))  
 *println*("Net Income after Tax: INR %.2f".*format*(finalIncome))  
}  
  
fun calculateIncomeTax(income: Double): Double {  
 return when {  
 income <= 400000 -> 0.0  
 income in 400001.0..800000.0 -> (income - 300000) \* 0.05  
 income in 800001.0..1200000.0 -> 20000 + (income - 800000) \* 0.10  
 income in 1200001.0..1600000.0 -> 60000 + (income - 1200000) \* 0.15  
 income in 1600001.0..2000000.0 -> 120000 + (income - 1600000) \* 0.20  
 income in 2000001.0..2400000.0 -> 200000 + (income - 2000000) \* 0.25  
 else -> 300000 + (income - 2400000) \* 0.30  
 }  
}

**Output:**Enter Gross Income (INR): 200000

Total Tax Deduction: INR 0.00

Net Income after Tax: INR 125000.00

**Question 2**

**Code:**fun calculateElectricityBill(units: Int): Double {  
 val fixedCharges = when {  
 units in 1..50 -> 230.0  
 units in 51..150 -> 230.0  
 units in 151..300 -> 275.0  
 units in 301..500 -> 345.0  
 else -> 400.0  
 }  
  
 var billAmount = fixedCharges  
 var remainingUnits = units  
  
 val slabs = *listOf*(  
 Triple(1, 50, 5.00),  
 Triple(51, 150, 6.50),  
 Triple(151, 300, 7.50),  
 Triple(301, 500, 7.85),  
 Triple(501, Int.MAX\_VALUE, 8.00)  
 )  
  
 for ((low, high, rate) in slabs) {  
 if (units > low) {  
 val chargeableUnits = *minOf*(high - low + 1, remainingUnits)  
 billAmount += chargeableUnits \* rate  
 remainingUnits -= chargeableUnits  
 }  
 if (remainingUnits <= 0) break  
 }  
  
 return billAmount  
}  
  
fun main() {  
 *print*("Enter number of units consumed: ")  
 val units = *readLine*()?.*toIntOrNull*() ?: return *println*("Invalid input")  
 val totalBill = *calculateElectricityBill*(units)  
 *println*("Electricity Bill Amount: INR %.2f".*format*(totalBill))  
 *println*("Number of Units Consumed: $units")  
}

**Output:**Enter number of units consumed: 30

Electricity Bill Amount: INR 380.00

Number of Units Consumed: 30

**Question 3**

**code:**fun isPrime(n: Int): Boolean {  
 if (n < 2) return false  
 if (n == 2 || n == 3) return true  
 if (n % 2 == 0 || n % 3 == 0) return false  
 var i = 5  
 while (i \* i <= n) {  
 if (n % i == 0 || n % (i + 2) == 0) return false  
 i += 6  
 }  
 return true  
}  
  
fun main() {  
 *print*("Enter a number: ")  
 val num = *readLine*()?.*toIntOrNull*() ?: return *println*("Invalid input")  
 val result = if (*isPrime*(num)) "a Prime Number" else "not a Prime Number"  
 *println*("$num is $result")  
}

**Output:**Enter a number: 2313

2313 is not a Prime Number

**Question 4**

**Code:**

fun sieveOfEratosthenes(start: Int, end: Int): List<Int> {  
 if (end < 2) return *emptyList*()  
 val isPrime = BooleanArray(end + 1) **{** true **}** isPrime[0] = false  
 isPrime[1] = false  
 for (i in 2..Math.sqrt(end.toDouble()).toInt()) {  
 if (isPrime[i]) {  
 for (j in i \* i..end *step* i) {  
 isPrime[j] = false  
 }  
 }  
 }  
 return (*maxOf*(2, start)..end).*filter* **{** isPrime[**it**] **}**}  
  
fun main() {  
 *print*("Enter the start of the range: ")  
 val start = *readLine*()?.*toIntOrNull*() ?: return *println*("Invalid input")  
 *print*("Enter the end of the range: ")  
 val end = *readLine*()?.*toIntOrNull*() ?: return *println*("Invalid input")  
  
 val primes = *sieveOfEratosthenes*(start, end)  
 *println*("Prime numbers in the range [$start, $end]: ${primes.*joinToString*(", ")}")  
}

**Output:**

Enter the start of the range: 2

Enter the end of the range: 41

Prime numbers in the range [2, 41]: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41

**Question 5**

**Code:**

fun isPerfectNumber(n: Int): Boolean {  
 if (n < 2) return false  
 var sum = 1  
 for (i in 2..n / 2) {  
 if (n % i == 0) sum += i  
 }  
 return sum == n  
}  
  
fun findPerfectNumbers(start: Int, end: Int): List<Int> {  
 return (start..end).*filter* **{** *isPerfectNumber*(**it**) **}**}  
  
fun main() {  
 *print*("Enter the start of the range: ")  
 val start = *readLine*()?.*toIntOrNull*() ?: return *println*("Invalid input")  
 *print*("Enter the end of the range: ")  
 val end = *readLine*()?.*toIntOrNull*() ?: return *println*("Invalid input")  
  
 val perfectNumbers = *findPerfectNumbers*(start, end)  
 if (perfectNumbers.isEmpty()) {  
 *println*("No perfect numbers found in the range [$start, $end].")  
 } else {  
 *println*("Perfect numbers in the range [$start, $end]: ${perfectNumbers.*joinToString*(", ")}")  
 }  
}

**Output:**

Enter the start of the range: 2

Enter the end of the range: 10

Perfect numbers in the range [2, 10]: 6

**Question 7**

**Code:**

fun countTriangles(sides: IntArray): Int {  
 sides.*sort*() // Sorting helps in using two-pointer approach  
 val n = sides.size  
 var count = 0  
  
 for (i in n - 1 *downTo* 2) { // Fix the largest side at i  
 var left = 0  
 var right = i - 1  
 while (left < right) {  
 if (sides[left] + sides[right] > sides[i]) {  
 count += (right - left) // All pairs (left to right-1) are valid  
 right--  
 } else {  
 left++  
 }  
 }  
 }  
 return count  
}  
  
fun main() {  
 *print*("Enter number of sides: ")  
 val n = *readLine*()!!.*toInt*()  
 *print*("Enter side lengths separated by space: ")  
 val sides = *readLine*()!!.*split*(" ").*map* **{ it**.*toInt*() **}**.*toIntArray*()  
  
 val result = *countTriangles*(sides)  
 *println*("Number of valid triangles: $result")  
}

**Output:**

Enter number of sides: 6

Enter side lengths separated by space: 4 6 3 7 8 10

Number of valid triangles: 14

**Question 8**

**Code:**import java.util.\*  
  
fun getDayOfWeek(day: Int, month: Int, year: Int): String {  
 val monthMagicNumbers = *intArrayOf*(0, 3, 3, 6, 1, 4, 6, 2, 5, 0, 3, 5)  
 val days = *arrayOf*("Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday")  
 val centuryCode = when (year / 100) {  
 16 -> 6  
 17 -> 4  
 18 -> 2  
 19 -> 0  
 20 -> 6  
 else -> 0  
 }  
  
 val lastTwoDigits = year % 100  
 val leapYearAdjustment = if (*isLeapYear*(year) && (month == 1 || month == 2)) -1 else 0  
  
 val sum = lastTwoDigits + (lastTwoDigits / 4) + day + monthMagicNumbers[month - 1] + centuryCode + leapYearAdjustment  
 val dayOfWeek = sum % 7  
  
 return days[dayOfWeek]  
}  
  
fun isLeapYear(year: Int): Boolean {  
 return (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)  
}  
  
fun main() {  
 val scanner = Scanner(System.*`in`*)  
  
 *print*("Enter day (dd): ")  
 val day = scanner.nextInt()  
  
 *print*("Enter month (mm): ")  
 val month = scanner.nextInt()  
  
 *print*("Enter year (yyyy): ")  
 val year = scanner.nextInt()  
  
 val result = *getDayOfWeek*(day, month, year)  
 *println*("The day of the week for $day/$month/$year is: $result")  
}

**Output:**Enter day (dd): 20

Enter month (mm): 4

Enter year (yyyy): 2025

The day of the week for 20/4/2025 is: Sunday

**Question 9**

**Code:**import java.util.\*  
  
fun findElements(arr: IntArray) {  
 var largest = Int.MIN\_VALUE  
 var secondLargest = Int.MIN\_VALUE  
 var smallest = Int.MAX\_VALUE  
 var secondSmallest = Int.MAX\_VALUE  
  
 for (num in arr) {  
 if (num > largest) {  
 secondLargest = largest  
 largest = num  
 } else if (num > secondLargest && num != largest) {  
 secondLargest = num  
 }  
  
 if (num < smallest) {  
 secondSmallest = smallest  
 smallest = num  
 } else if (num < secondSmallest && num != smallest) {  
 secondSmallest = num  
 }  
 }  
  
 *println*("Largest: $largest")  
 *println*("Second Largest: $secondLargest")  
 *println*("Smallest: $smallest")  
 *println*("Second Smallest: $secondSmallest")  
}  
  
fun main() {  
 val scanner = Scanner(System.*`in`*)  
 *print*("Enter the number of elements: ")  
 val n = scanner.nextInt()  
 val arr = IntArray(n)  
  
 *println*("Enter the elements:")  
 for (i in 0 *until* n) {  
 arr[i] = scanner.nextInt()  
 }  
  
 *findElements*(arr)  
}

**Output:**Enter the number of elements: 6

Enter the elements:

4 1 8 3 7 2

Largest: 8

Second Largest: 7

Smallest: 1

Second Smallest: 2