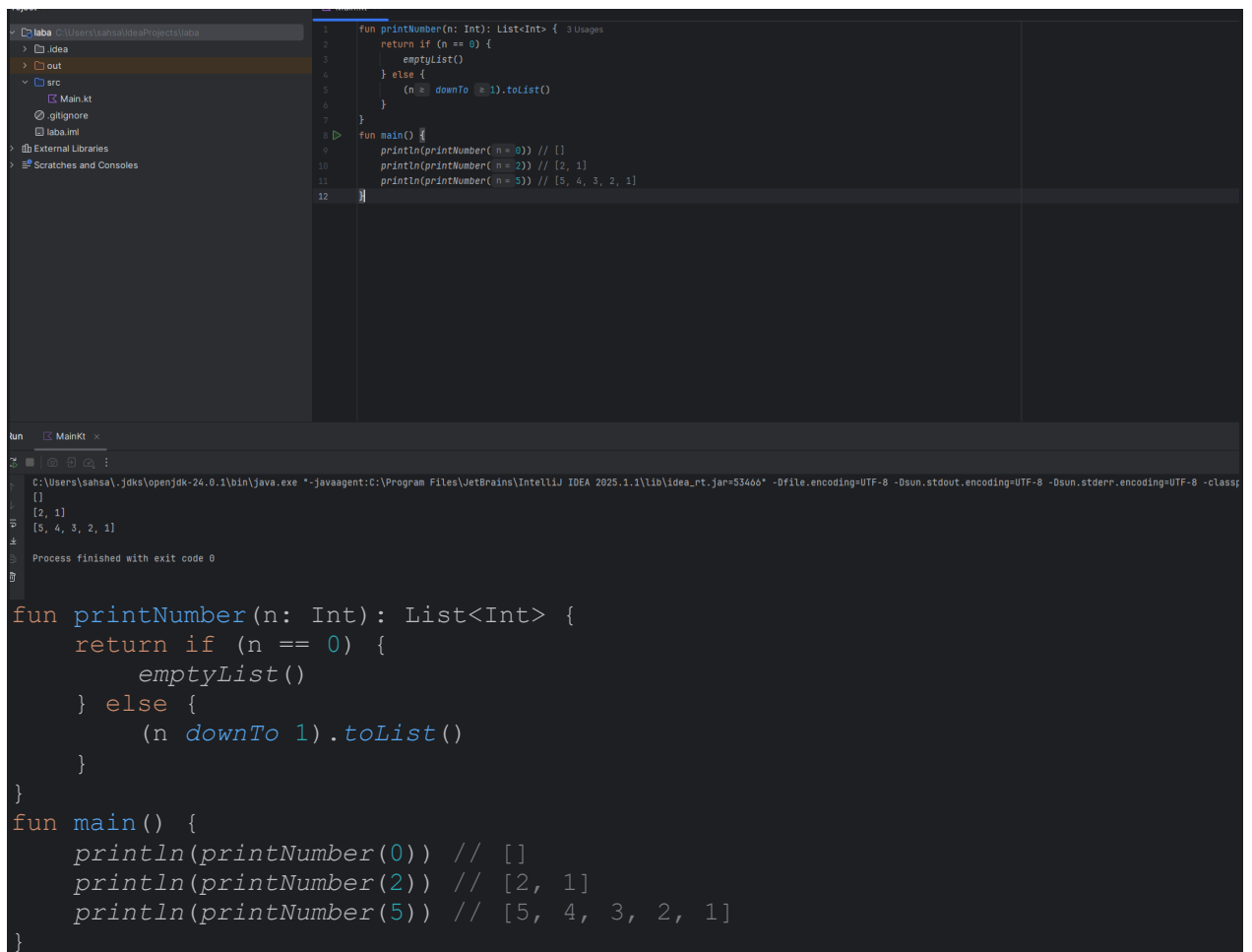


Лабораторная работа №10

Харахардин А. ИС233

1.



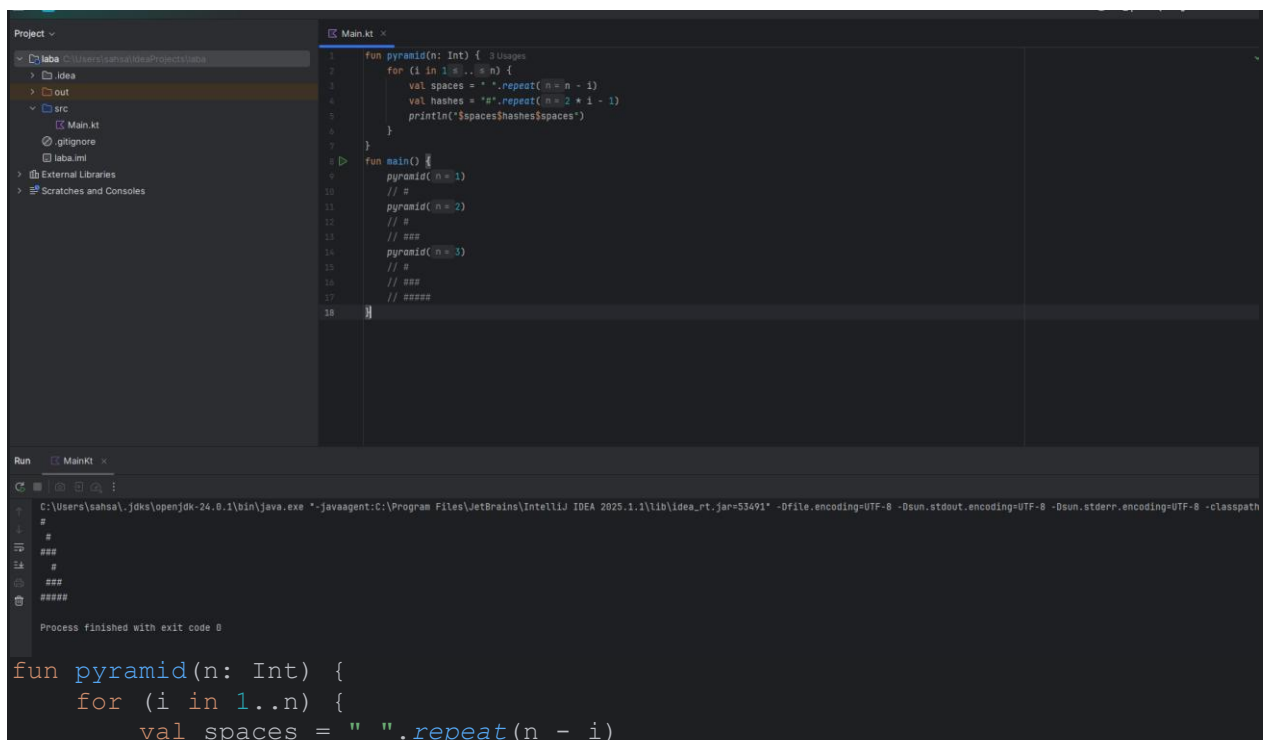
The screenshot shows the IntelliJ IDEA IDE with a Kotlin file named `Main.kt`. The code defines a function `printNumber` that returns a list of integers for a given `n`. The `main` function calls `printNumber` with arguments 0, 2, and 5, and prints the results. The Run window shows the output: `[]`, `[2, 1]`, and `[5, 4, 3, 2, 1]`. Below the Run window, the source code is displayed again.

```
1 fun printNumber(n: Int): List<Int> { 3 Usages
2     return if (n == 0) {
3         emptyList()
4     } else {
5         (n downTo 1).toList()
6     }
7 }
8 fun main() {
9     println(printNumber(0)) // []
10    println(printNumber(2)) // [2, 1]
11    println(printNumber(5)) // [5, 4, 3, 2, 1]
12 }
```

```
run C:\Users\sahsa\.jdk\openjdk-24.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.1.1\lib\idea_rt.jar=53466" -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8 -classpath
[ ]
[2, 1]
[5, 4, 3, 2, 1]
Process finished with exit code 0
```

```
fun printNumber(n: Int): List<Int> {
    return if (n == 0) {
        emptyList()
    } else {
        (n downTo 1).toList()
    }
}
fun main() {
    println(printNumber(0)) // []
    println(printNumber(2)) // [2, 1]
    println(printNumber(5)) // [5, 4, 3, 2, 1]
}
```

2.



The screenshot shows the IntelliJ IDEA IDE with a Kotlin file named `Main.kt`. The code defines a function `pyramid` that prints a pyramid of spaces and hashes for a given `n`. The `main` function calls `pyramid` with arguments 1, 2, and 3, and prints the results. The Run window shows the output: `#`, `##`, `###`, `####`, and `#####`. Below the Run window, the source code is displayed again.

```
1 fun pyramid(n: Int) { 3 Usages
2     for (i in 1..n) {
3         val spaces = " ".repeat(n - i)
4         val hashes = "#".repeat(i)
5         println("${spaces}${hashes}${spaces}")
6     }
7 }
8 fun main() {
9     pyramid(1)
10    // #
11    pyramid(2)
12    // #
13    // ###
14    pyramid(3)
15    // #
16    // ###
17    // #####
18 }
```

```
run C:\Users\sahsa\.jdk\openjdk-24.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.1.1\lib\idea_rt.jar=53491" -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8 -classpath
#
##
###
####
#####
Process finished with exit code 0
```

```
fun pyramid(n: Int) {
    for (i in 1..n) {
        val spaces = " ".repeat(n - i)
```

```

        val hashes = "#".repeat(2 * i - 1)
        println("$spaces$hashes$spaces")
    }
}
fun main() {
    pyramid(1)
    // #
    pyramid(2)
    // #
    // ###
    pyramid(3)
    // #
    // ###
    // #####
}

```

3.

```

1 fun caesarCipher(text: String, shift: Int): String {
2     return text.map { char ->
3         when {
4             char.isLowerCase() -> {
5                 val start = 'a'.code
6                 val shifted = (char.code - start + shift) % 32 + start
7                 shifted.toChar()
8             }
9             char.isUpperCase() -> {
10                val start = 'A'.code
11                val shifted = (char.code - start + shift) % 32 + start
12                shifted.toChar()
13            }
14            else -> char
15        }
16    }.joinToString(separator = "")
17 }
18 fun main() {
19     val text = "Привет, мир!"
20     val shift = 3
21     val encrypted = caesarCipher(text, shift)
22     println("Зашифрованный текст: $encrypted")
23     val decrypted = caesarCipher(encrypted, -shift)
24     println("Расшифрованный текст: $decrypted")
25 }

```

```

fun caesarCipher(text: String, shift: Int): String {
    return text.map { char ->
        when {
            char.isLowerCase() -> {
                val start = 'a'.code
                val shifted = (char.code - start + shift) % 32 + start
                shifted.toChar()
            }
            char.isUpperCase() -> {
                val start = 'A'.code
                val shifted = (char.code - start + shift) % 32 + start
                shifted.toChar()
            }
            else -> char
        }
    }.joinToString("")
}
fun main() {
    val text = "Привет, мир!"
    val shift = 3
    val encrypted = caesarCipher(text, shift)
    println("Зашифрованный текст: $encrypted")
    val decrypted = caesarCipher(encrypted, -shift)
    println("Расшифрованный текст: $decrypted")
}

```

4.

The screenshot shows an IDE with a project named 'Idea'. The source code is in 'Main.kt' and implements a FizzBuzz function in Kotlin. The function 'fizzBuzz' takes an integer 'n' and returns a list of strings. It uses a 'when' expression to handle divisibility by 15, 3, and 5. The 'main' function calls 'fizzBuzz' with arguments 5 and 16, and prints the results. The output window shows the execution of the program, displaying the list of strings for n=5 and n=16. The process finished with exit code 0.

```
1 fun fizzBuzz(n: Int): List<Any> { 2 Usages
2     return (1..n).map {
3
4         when {
5             it % 15 == 0 -> " ВизллБизлл "
6             it % 3 == 0 -> " Физллл "
7             it % 5 == 0 -> " Бизлллл "
8             else -> it
9         }
10    }
11 }
12 fun main() {
13     println(fizzBuzz(5))
14     println(fizzBuzz(16))
15 }
```

Process finished with exit code 0

```
fun fizzBuzz(n: Int): List<Any> {
    return (1..n).map {
        when {
            it % 15 == 0 -> " ВизллБизлл "
            it % 3 == 0 -> " Физллл "
            it % 5 == 0 -> " Бизлллл "
            else -> it
        }
    }
}

fun main() {
    println(fizzBuzz(5))
    println(fizzBuzz(16))
}
```