**Кросплатформенне програмування**

## **Лабораторна робота №2**

«Основи функціонального програмування у Kotlin»

*Виконав Заяць Ілля, група ІПЗ-41*

**Мета:**

Виконати всі завдання із використанням інформації з лекцій "Колекції" та "Функціональне програмування".

Хід роботи

1.)- Повернути список квадратів даного списку чисел.

1а.)- Повернути суму квадратів даного списку чисел.

2.)- Маючи два списки чисел, повернути список чисел з поєднаних пар чисел з кожного списка.

3.)- Виконати велике завдання, доповнивши код (детальніше нижче).

## *Завдання 1*

1.)- Маючи список чисел, як би ви повернули список квадратів кожного

числа? Наприклад, якщо задано [1, 2, 3, 4, 5], ви повинні повернути

[1, 4, 9, 16, 25]. Є дві змінні – val numbers, яка відповідає за задання самого масиву/списку чисел, і val squares, яка як раз повертає список квадратів заданих чисел.

2.)- Код:

fun main() {

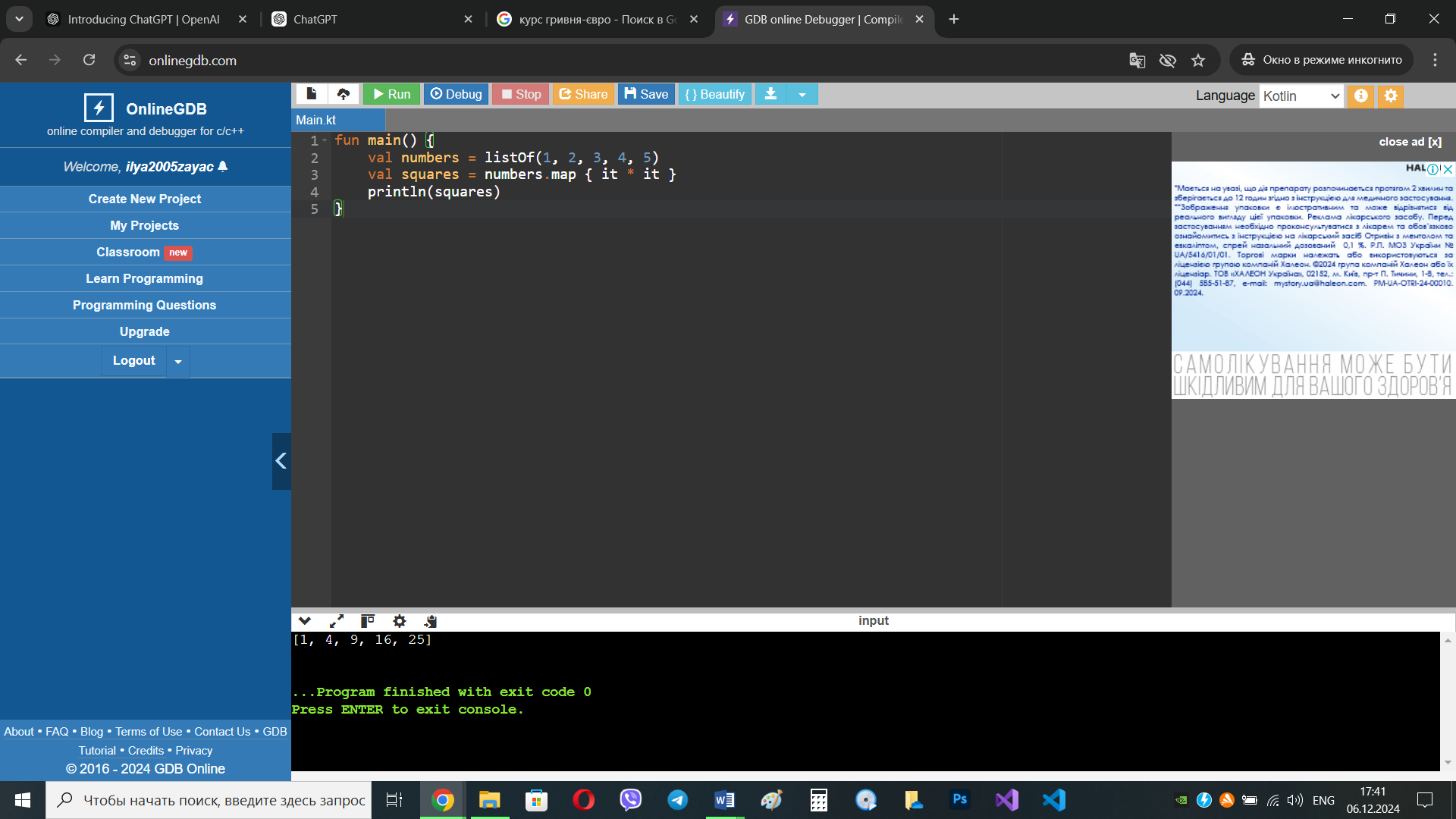
val numbers = listOf(1, 2, 3, 4, 5)

val squares = numbers.map { it \* it }

println(squares)

}

3.)- Результат виконання (онлайн-компілятор online gdb):



## *Завдання 1а*

1.)- Знайти суму квадратів даного списку. Доповнення програми з завдання 1. Друга змінна тепер називається sumOfSquares, і вона повертає вже суму квадратів заданого списку.

2.)- Код:

fun main() {

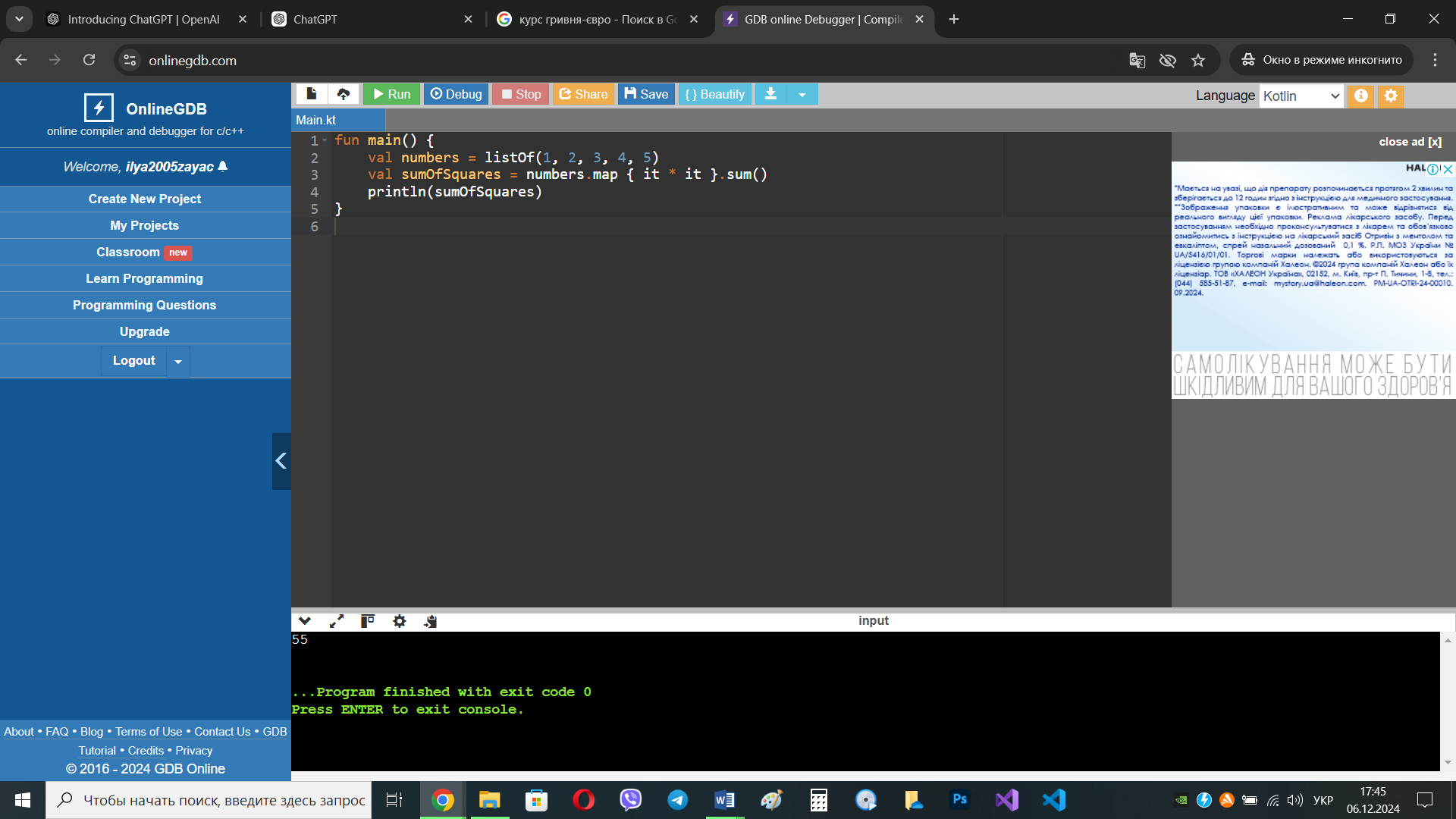
val numbers = listOf(1, 2, 3, 4, 5)

val sumOfSquares = numbers.map { it \* it }.sum()

println(sumOfSquares)

}

3.)- Результат виконання (онлайн-компілятор online gdb):



## *Завдання 2*

1.)- Дано два списки чисел, як би ви повернули всі пари чисел? Наприклад, враховуючи список [1, 2, 3] і список [3, 4], ви повинні повернути [(1, 3), (1, 4), (2, 3), (2, 4), (3, 3), (3, 4)]. Для простоти ви можете представити пару у вигляді масиву з двох елементів. Є три змінні: перші дві задають списки чисел, а третя, pairs, якраз і повертає новий список чисел, зіставлений з чисел з двох різних списків, поданих парами. Для запобігання помилки коду була додана строка pairs.forEach { println(it.joinToString(", ")) }, щоб програма виводила пари чисел окремо, взаємодіючи лише з самими значеннями в списках. Без цього виправлення вона намагалася працювати з самими масивами, що призводило до помилкового відображення.

2.)- Код:

fun main() {

val list1 = listOf(1, 2, 3)

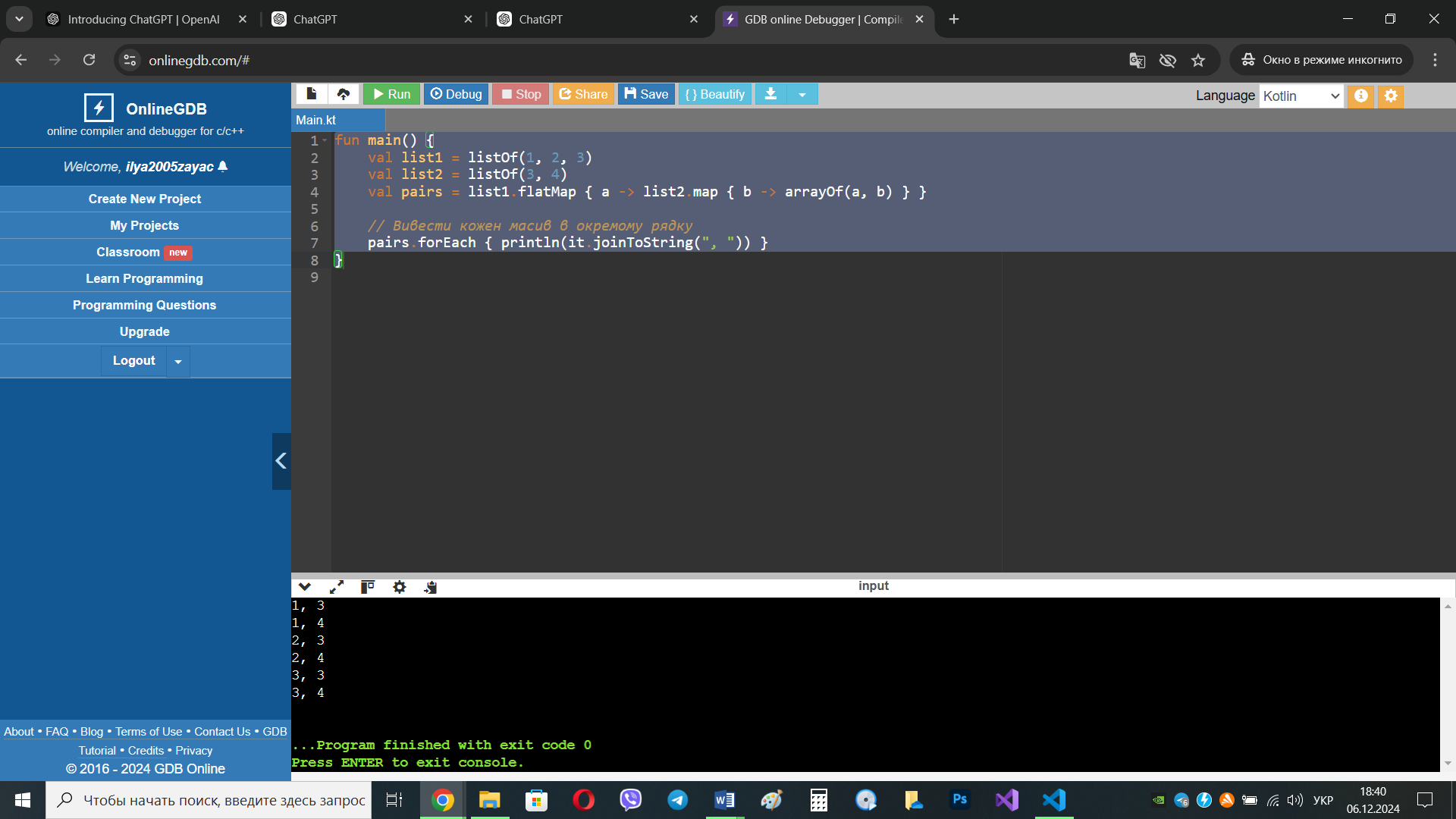
val list2 = listOf(3, 4)

val pairs = list1.flatMap { a -> list2.map { b -> arrayOf(a, b) } }

pairs.forEach { println(it.joinToString(", ")) }

}

3.)- Результат виконання (онлайн-компілятор online gdb):



## *Завдання 3*

## 1.)- Велике завдання. Завдання будуть наведені нижче, і відразу ж буде розписане рішення.

*1. Знайти усі транзакції за 2011 рік і посортувати за вартістю (від малого до високого).*

Треба відфільтрувати транзакції, де рік = 2011, і відсортувати їх за полем value (вартість).

*2. У яких унікальних містах працюють трейдери?*

Я використав map для отримання списку міст і distinct для отримання унікальних міст. Результат: Кембридж, Мілан.

*3. Знайдіть усіх трейдерів із Кембриджа та відсортуйте їх за назвою*

Я відфільтрував трейдерів, що живуть у Кембриджі, відібрав унікальних і відсортував за іменем.

*4. Поверніть рядок імен усіх трейдерів, відсортованих за алфавітом*

Я використав distinct для унікальних імен, відсортував їх і перетворив в рядок через joinToString.

*5. Чи є трейдери в Мілані?*

Використав any, щоб перевірити наявність хоча б одного трейдера в Мілані. Так, в Мілані є один трейдер.

*6. Виведіть у консоль всі значення транзакцій від трейдерів, які проживають у Кембриджі*

Треба відфільтрувати транзакції, де трейдер з Кембриджа, і вивести результат.

*7. Знайдіть транзакцію з найбільшою вартістю.*

Я використав maxByOrNull для пошуку транзакції з найбільшою вартістю.

*8. Згрупуйте всі транзакції за валютою.*

Використаний groupBy для групування транзакцій за валютою.

*9. Знайдіть суму транзакцій у гривнях.*

Для кожної транзакції перетворюємо значення в гривні за допомогою курсу обміну. Якщо вірити Інтернету, зараз курс такий: 1 євро = 43,84 гривні; 1 долар = 41,42 гривні. Однак формат змінної не підтримує дійсні числа, тому округлимо до 44 і 41 гривень відповідно.

*10. Створіть рядок, у якому буде виведена послідовність транзакцій відсортована за датою у наступному вигляді (<назва параметру, який потрібно вставити> ):*

*«<numOfCurrency>. <month>-<year>: <value> <currency> -> <next*

*transaction>».*

*Наприклад:*

*«1. 11-2023: 2000 UAH -> 2. 12-2023 300 USD -> 3. 1-2024 400*

*EUR».*

Сортовано транзакції за датою та створений рядок, що містить всі транзакції у вказаному форматі.

## 2.)- Код. У самий код додані коментарі, щоб візуально розділити код різних завдань:

import Currency.EUR

import Currency.UAH

import Currency.USD

data class Trader(val name: String, val city: String)

data class Transaction(val trader: Trader, val year: Int, val month: Int, val value: Int, val currency: Currency)

enum class Currency {

UAH, USD, EUR

}

fun main() {

val raoul = Trader("Raoul", "Cambridge")

val mario = Trader("Mario", "Milan")

val alan = Trader("Alan", "Cambridge")

val brian = Trader("Brian", "Cambridge")

val transactions = listOf(

Transaction(brian, 2011, 12, 300, UAH),

Transaction(raoul, 2012, 10, 1000, UAH),

Transaction(raoul, 2011, 11, 400, USD),

Transaction(mario, 2012, 9, 710, UAH),

Transaction(mario, 2012, 7, 700, USD),

Transaction(alan, 2012, 4, 950, EUR)

)

// 1. Знайти усі транзакції за 2011 рік і посортувати за вартістю (від малого до високого)

val transactions2011 = transactions.filter { it.year == 2011 }

.sortedBy { it.value }

println("Transactions from 2011 sorted by value: $transactions2011")

// 2. У яких унікальних містах працюють трейдери?

val uniqueCities = transactions.map { it.trader.city }.distinct()

println("Unique cities where traders work: $uniqueCities")

// 3. Знайдіть усіх трейдерів із Кембриджа та відсортуйте їх за назвою

val tradersInCambridge = transactions.filter { it.trader.city == "Cambridge" }

.map { it.trader }

.distinct()

.sortedBy { it.name }

println("Traders in Cambridge sorted by name: $tradersInCambridge")

// 4. Поверніть рядок імен усіх трейдерів, відсортованих за алфавітом

val sortedTradersNames = transactions.map { it.trader.name }

.distinct()

.sorted()

.joinToString(", ")

println("Traders names sorted alphabetically: $sortedTradersNames")

// 5. Чи є трейдери в Мілані?

val tradersInMilan = transactions.any { it.trader.city == "Milan" }

println("Are there any traders in Milan? $tradersInMilan")

// 6. Виведіть у консоль всі значення транзакцій від трейдерів, які проживають у Кембриджі

val transactionsInCambridge = transactions.filter { it.trader.city == "Cambridge" }

println("Transactions from traders in Cambridge: $transactionsInCambridge")

// 7. Знайдіть транзакцію з найбільшою вартістю.

val maxTransaction = transactions.maxByOrNull { it.value }

println("Transaction with the highest value: $maxTransaction")

// 8. Згрупуйте всі транзакції за валютою.

val transactionsGroupedByCurrency = transactions.groupBy { it.currency }

println("Transactions grouped by currency: $transactionsGroupedByCurrency")

// 9. Знайдіть суму транзакцій у гривнях.

// Для цього потрібно врахувати курс обміну. Ми припускаємо, що:

// 1 USD = 41 UAH

// 1 EUR = 44 UAH

val totalInUAH = transactions.sumOf { transaction ->

when (transaction.currency) {

UAH -> transaction.value

USD -> transaction.value \* 41

EUR -> transaction.value \* 44

}

}

println("Total transactions in UAH: $totalInUAH")

// 10. Створіть рядок, у якому буде виведена послідовність транзакцій відсортована за датою

val sortedTransactions = transactions.sortedWith(compareBy({ it.year }, { it.month }))

val transactionSequence = sortedTransactions

.mapIndexed { index, transaction ->

"${index + 1}. ${transaction.month}-${transaction.year}: ${transaction.value} ${transaction.currency} "

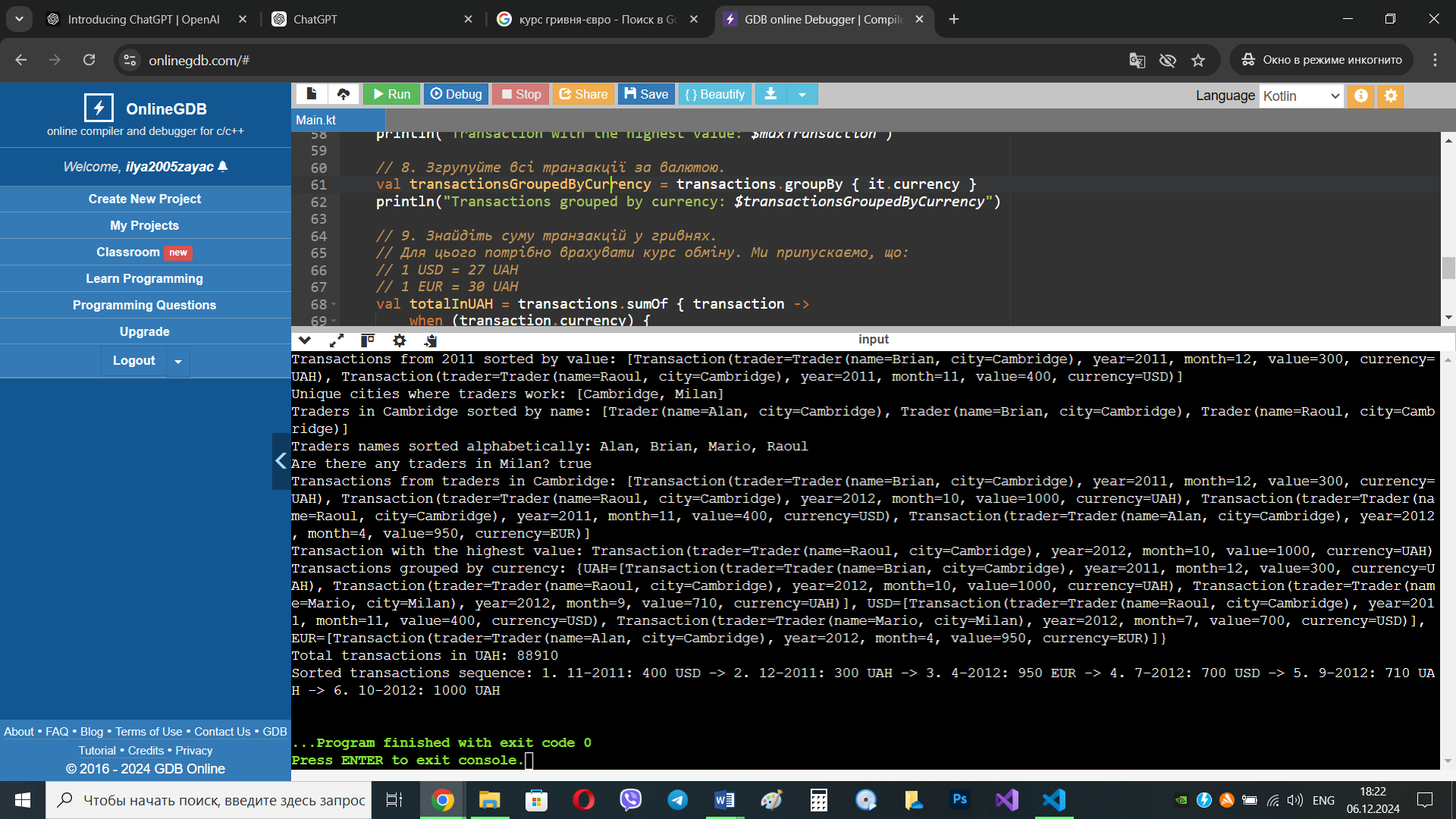
}

.joinToString("-> ")

println("Sorted transactions sequence: $transactionSequence")

}

## 3.)- Результат виконання (онлайн-компілятор online gdb):



## **Висновок:**

Перші три завдання (якщо враховувати 1а) було досить просто зробити, і про них мало що можна сказати. А ось завдання 2 було значно складніше, однак я впорався. Програма виконує всі поставлені завдання, при цьому залишаючись легкою для розуміння, за рахунок коментарів.