**Міністерство освіти і науки України**

**Національний університет «Запорізька Політехніка»**

Кафедра програмних засобів

**ЗВІТ**

з лабораторної роботи №1

з дисципліни «Спортивне програмування» на тему:

«Рекурентні послідовності»

**Виконав:**

Студент групи КНТ-122 О. А. Онищенко

**Прийняли:**

Викладач: С. Д. Леощенко

2023

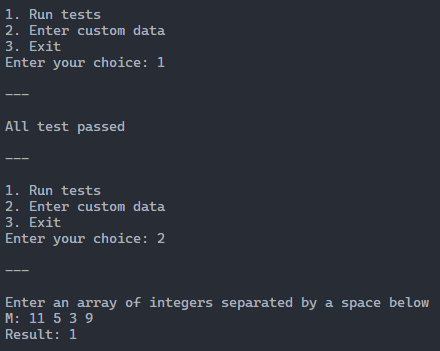
**Рекурентні послідовності**

**Мета роботи**

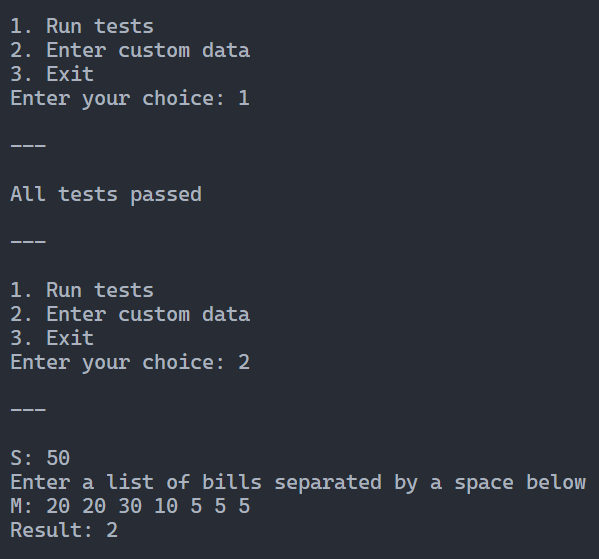
Вивчити основні можливості та принципи роботи з мовою рекурентні послідовності та співвідношення.

**Завдання до роботи**

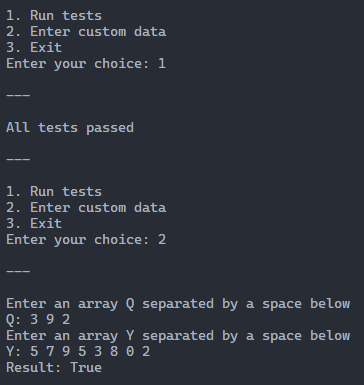
**Результати виконання**



Результати роботи програми 1



Результати роботи програми 3



Результат роботи програми 9

**Програмний код**

"""

Задано масив M[1: N] натуральних чисел, упорядкований за зростанням. Знайти перше натуральне число, яке не можна представити сумою жодних елементів масиву, при цьому сума може складатися і з одного доданка, але кожен елемент масиву може входити в неї лише один раз.

"""

def solve(*M*: [int]) -> int:

*# for keeping track of current sum*

    sum = 1

*# loop over each item in the given array*

    for item in *M*:

*# check if a current item is bigger than the sum*

*# this would mean that either the sum can't be created by any elements in the array, so its too small, or that the sum we have is larger than the sum of all the elements in the array. in any case, we have the number we're looking for*

        if item > sum:

*# so we just break out of the loop*

            break

*# in any other case*

        else:

*# just add the current element to our sum holder*

            sum += item

*# and return the number we got*

    return sum

def tests():

    assert solve([1, 2, 4]) == 8, "Test 1 failed"

    assert solve([1, 3, 6, 10]) == 2, "Test 2 failed"

    assert solve([2, 5, 11]) == 1, "Test 3 failed"

    print("All test passed")

def main():

    print()

    while True:

        print("1. Run tests")

        print("2. Enter custom data")

        print("3. Exit")

        choice = input("Enter your choice: ")

        print("\n---\n")

        if choice == "1":

            tests()

        elif choice == "2":

            print("Enter an array of integers separated by a space below")

            data = list(map(int, input("M: ").split()))

            print(f"Result: {solve(data)}")

        else:

            break

        print("\n---\n")

if \_\_name\_\_ == "\_\_main\_\_":

    main()

"""

Задано масив M[1: N] натуральних чисел, упорядкований за зростанням. Написати алгоритм виплати заданої суми S мінімальною кількістю купюр гідністю M(1), … , M(N).

"""

def solve(*M*: [int], *S*: int) -> int:

*# checking if either our cash register is empty, or we need no change to give*

    if len(*M*) == 0 or *S* == 0:

*# in either case, just return 0*

        return 0

*# for sorting and aligning the bills array to make sure all the bills there are less than or equal to our target*

    def alignBills(*arr*, *target*):

        return sorted([item for item in *arr* if item <= *target*])

*# align the initial bills with a target sum*

    existingBills = alignBills(*M*, *S*)

*# for keeping track of how much more do we need to give in change*

    leftToPay = *S*

*# for keeping track of the number of bills we've used*

    requiredNumberOfBills = 0

*# while we still have some change to give*

    while leftToPay >= 0:

*# align bills so we get only the ones that are smaller than the one we need to pay out*

        existingBills = alignBills(existingBills, leftToPay)

*# for chosing an optimal bill*

        chosenBill = 0

*# check if the amount that is left to pay is in the array*

        if leftToPay in existingBills:

*# in such case, choose this bill and remove it from the cash register*

            chosenBill = leftToPay

            existingBills.remove(chosenBill)

*# if its not*

        else:

*# then just remove the largest one*

            chosenBill = existingBills.pop()

*# decrease the amount we have left to pay and increase the amount of bills we have payed so far*

        leftToPay -= chosenBill

        requiredNumberOfBills += 1

*# check if we need no more to pay*

        if leftToPay == 0:

*# if so, break out*

            break

    return requiredNumberOfBills

def tests():

    assert solve([1, 2, 5, 10, 20, 50], 71) == 3, "Test 1 failed"

    assert solve([1, 2, 5, 10, 20, 50], 3) == 2, "Test 2 failed"

    assert solve([1, 2, 5, 10, 20, 50], 0) == 0, "Test 3 failed"

    assert solve([1, 2, 5, 10, 20, 50], 50) == 1, "Test 4 failed"

    assert solve([1, 3, 4], 4) == 1, "Test 5 failed"

    assert solve([100, 100, 20, 50, 5, 30, 50, 1, 1, 1, 5], 25) == 2, "Test 6 failed"

    print(f"All tests passed")

def main():

    print()

    while True:

        print("1. Run tests")

        print("2. Enter custom data")

        print("3. Exit")

        choice = input("Enter your choice: ")

        print("\n---\n")

        if choice == "1":

            tests()

        elif choice == "2":

            givenSum = int(input("S: "))

            print("Enter a list of bills separated by a space below")

            data = list(map(int, input("M: ").split()))

            print(f"Result: {solve(data,givenSum)}")

        else:

            break

        print("\n---\n")

if \_\_name\_\_ == "\_\_main\_\_":

    main()

"""

Задані Q та Y – дві послідовності. Чи можна отримати послідовність Q шляхом викреслення елементів з Y?

"""

def solve(*Q*: [int], *Y*: [int]) -> bool:

*# sort two arrays*

    takeFrom, lookIn = sorted(*Q*), sorted(*Y*)

*# for keeping track of the items in the original array*

    count = 0

*# loop over each item in the array*

    for item in takeFrom:

*# check if the item is in our target array*

        if item in lookIn:

*# if so, increment the counter*

            count += 1

*# if its not, it means that we cannot get an array Q from an array Y*

        else:

*# so we immediately return false*

            return False

    return True if count == len(takeFrom) else False

def tests():

    assert solve([1, 2, 3], [1, 2, 3, 4, 5]) == True, "Test 1 failed"

    assert solve([1, 2, 3], [1, 2]) == False, "Test 2 failed"

    assert solve([], [1, 2, 3]) == True, "Test 3 failed"

    assert solve([1, 2, 3], []) == False, "Test 4 failed"

    assert solve([1, 2, 3], [3, 2, 1]) == True, "Test 5 failed"

    print("All tests passed")

def main():

    print()

    while True:

        print("1. Run tests")

        print("2. Enter custom data")

        print("3. Exit")

        choice = input("Enter your choice: ")

        print("\n---\n")

        if choice == "1":

            tests()

        elif choice == "2":

            print("Enter an array Q separated by a space below")

            takeFrom = list(map(int, input("Q: ").split()))

            print("Enter an array Y separated by a space below")

            lookIn = list(map(int, input("Y: ").split()))

            print(f"Result: {solve(takeFrom,lookIn)}")

        else:

            break

        print("\n---\n")

if \_\_name\_\_ == "\_\_main\_\_":

    main()

**Висновки**

Таким чином, ми вивчили основні можливості та принципи роботи з мовою рекурентних послідовностей та співвідношень.