**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО**

Факультет інформатики та обчислювальної техніки Кафедра інформатики та програмної інженерії

Звіт з комп’ютерного практикуму №4

«Оцінка точності та складності алгоритму імітації.»

роботи з дисципліни: « Моделювання систем »

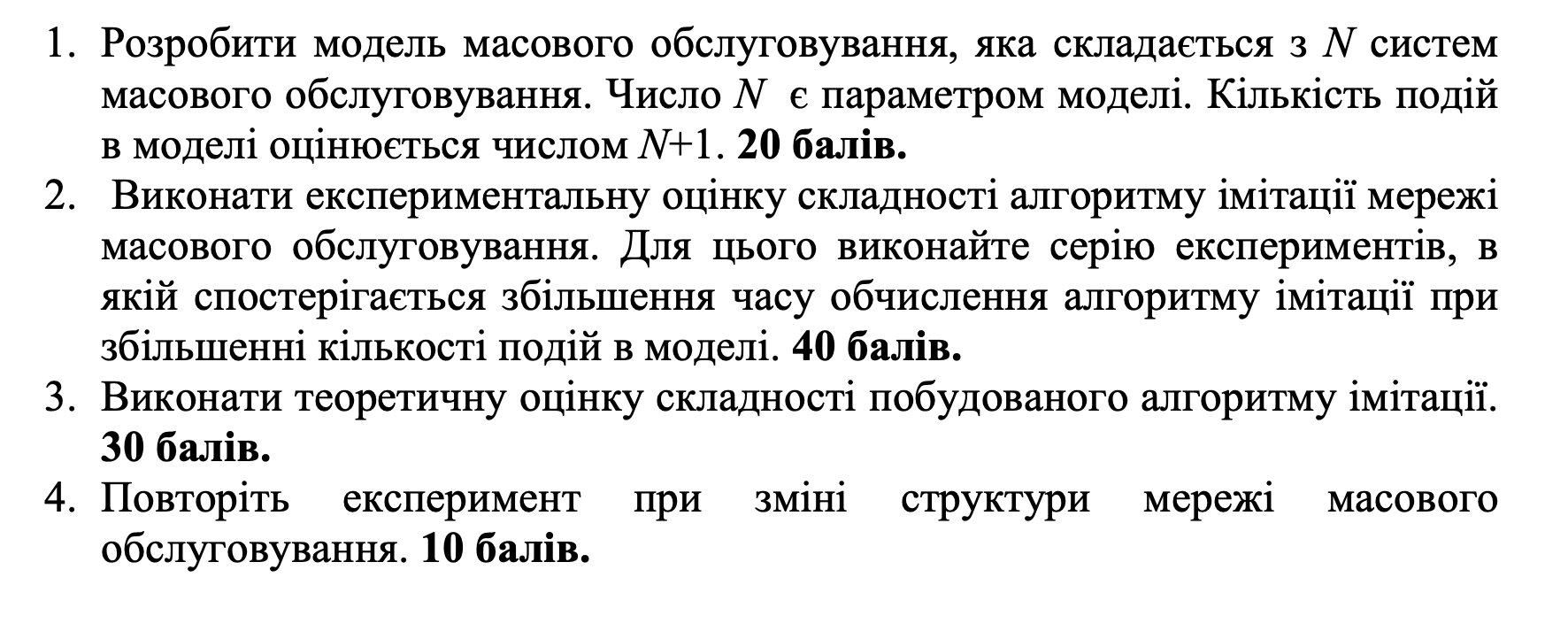
Студент: Мєшков Андрій Ігорович\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Група: ІП-15\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Викладач: асистент Дифучин А. Ю.\_\_\_\_\_\_\_\_\_\_

Київ, 2024

# Завдання

****

# Хід роботи

1. Створимо модель з n послідовно з’єднаних СМО.

func task1(\_ n: Int) -> Model {

let create = Create(name: "create", delay: 3)

create.distribution = .exp

var elements: [Element] = [create]

*for* index *in* 0..<n {

let process = Process(name: "process\(index)", delays: [1])

*if* elements.isEmpty {

create.transfer = SoloTransfer(nextElement: process)

} *else* {

elements.last?.transfer = SoloTransfer(nextElement: process)

}

process.queue = Queue(maxLength: .max)

process.distribution = .exp

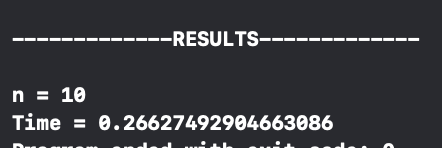
elements.append(process)

}

*return* Model(elements: elements, resultsPrinter: lab4Printer)

}

1. Проведемо тестування та експериментально визначимо часову складність.



Результат 4.1 - Результат

Результат 4.2 – Графік залежності часу роботи імітації від кількості подій

1. Теоретична оцінка складності

Теоретичну складність алгоритму можна описати як O(λ⋅T⋅ω), де:

* λ— частота появи подій (для генератора подій приймається рівною 1),
* T — загальний час симуляції,
* ω— середня кількість базових операцій для обробки однієї події.

Спрощуючи цей вираз, отримуємо O(1⋅T⋅N)=O(N), що свідчить про лінійну залежність складності від кількості подій. Експериментальні дані, представлені у вигляді графіків, підтверджують цю теоретичну оцінку, демонструючи лінійне зростання складності.

1. Створимо модель, що з останнього СМО з вірогідністю 0.5 надсилає задачі назад.

func task4(\_ n: Int) -> Model {

let create = Create(name: "create", delay: 3)

create.distribution = .exp

var elements: [Element] = [create]

*for* index *in* 0..<n {

let process = Process(name: "process\(index)", delays: [1])

*if* elements.isEmpty {

create.transfer = SoloTransfer(nextElement: process)

} *else* {

elements.last?.transfer = SoloTransfer(nextElement: process)

}

process.queue = Queue(maxLength: .max)

process.distribution = .exp

elements.append(process)

}

elements.last?.transfer = CustomTransfer(nextElement: { \_ *in*

*if* Double.random(in: 0..<1) > 0.5 {

*return* elements[1]

} *else* {

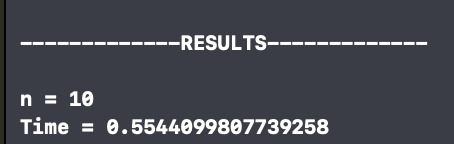
*return* nil

}

})

*return* Model(elements: elements, resultsPrinter: lab4Printer)

}



Результат 4.3 - Результат

Результат 4.4 – Графік залежності часу роботи імітації від кількості подій

Бачимо лінійну залежність.

# ВИСНОВКИ

У результаті виконання практичної роботи було створено 2 прості моделі, кількість СМО в яких залежить від параметра n. Було експериментально досліджено часову складність алгоритмів – O(n). Та підтверджено її з теоретичної точки зору.