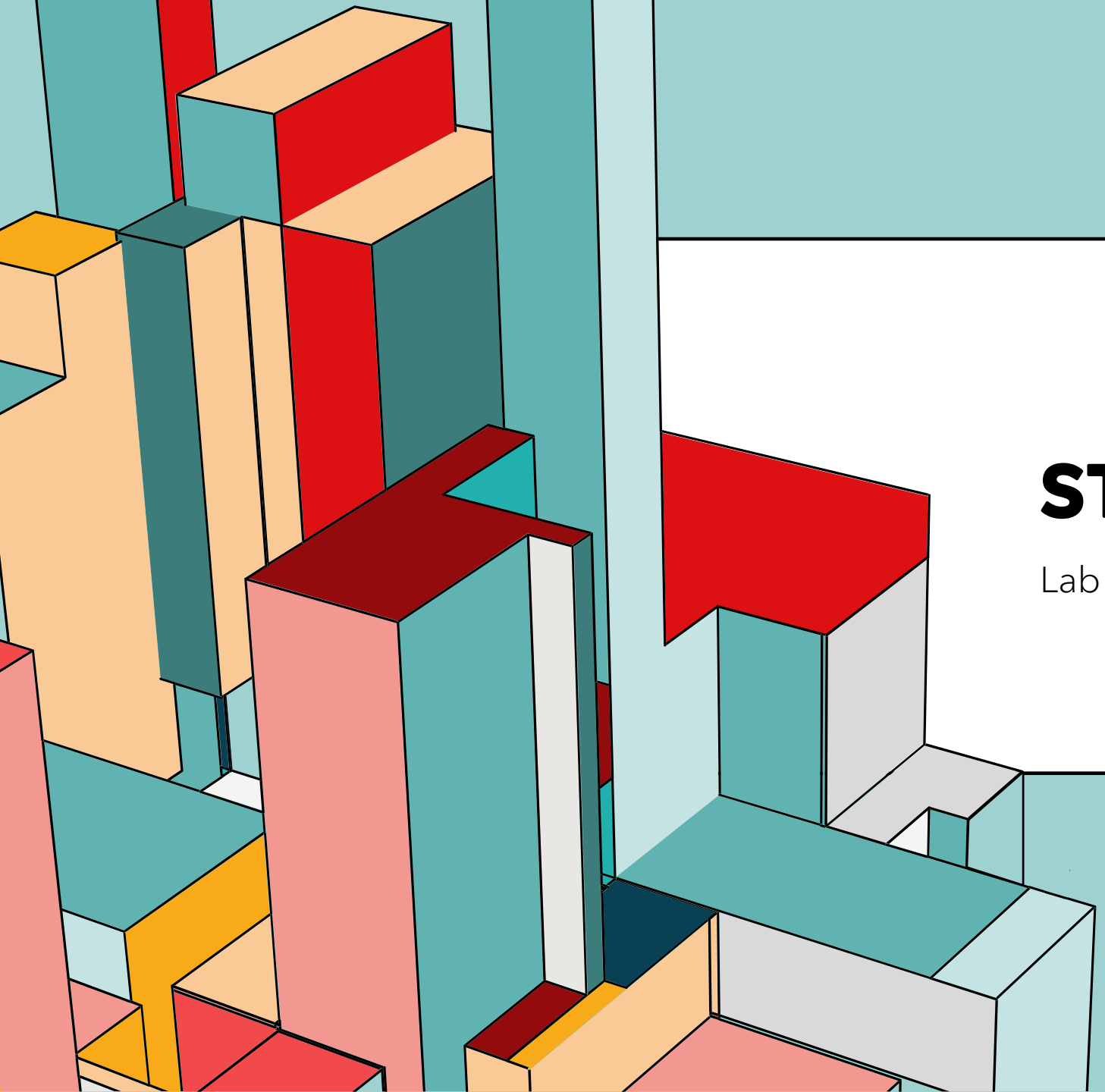

STACK

Lab Document (Lab 3-4)

LAB 4.1

- จงเขียนฟังก์ชัน **is_parentheses_matching(expression)** ที่รับสตริงเก็บค่า expression เช่น ((A+B)*C) และทำการคืนค่า (return) ผลลัพธ์เป็นค่า True หรือ False
  - คืนค่า True ก็ต่อเมื่อ expression มีการจับคู่ ‘(’ และ ‘)’ ครบคู่ เช่น ((A+B)*C)
  - ไม่เช่นนั้น ให้คืนค่า False เช่น (((A-B)*C)
- ตัวอย่าง
  - str = “(((A-B)*C)”
  - result = is_parentheses_matching(str)
    - โปรแกรมแสดง ***Parentheses in (((A+B)*C) are unmatched***
  - print(result) => โปรแกรมแสดง False
- **หมายเหตุ** กำหนดให้สร้าง Stack จากโครงสร้าง Lab 3 และเรียกใช้เมธอดของคลาส ArrayStack เป็นหลัก เช่น push() และ pop() เป็นต้น

LAB 4.2

- จงเขียนฟังก์ชัน **copyStack(stack1, stack2)** ที่รับอ็อบเจ็ค stack1 และ stack2 โดยทำการคัดลอกค่าข้อมูลใน stack1 ให้กับ stack2 โดยมีลำดับของข้อมูลเหมือนกัน
- ตัวอย่าง
  - s1 = ArrayStack(); s1.push(10); s1.push(20); s1.push(30) // s1 = [10,20,30]
  - s2 = ArrayStack(); s2.push(15); // s2 = [15]
  - copyStack(s1, s2)
  - s1.printStack() => โปรแกรมแสดง [10,20,30]
  - s2.printStack() => โปรแกรมแสดง [10,20,30]

- **หมายเหตุ** กำหนดให้สร้าง Stack จากโครงสร้าง Lab 3 และเรียกใช้เมธอดของคลาส ArrayStack เป็นหลัก เช่น push() และ pop() เป็นต้น

LAB 4.3

- จงเขียนฟังก์ชัน **infixToPostfix(expression)** ที่รับสตริงเก็บค่า expression เช่น A+B*C และทำการคืนค่า (return) ผลลัพธ์เป็นสตริง Postfix
  - กำหนดให้มีโอเปอเรเตอร์ ‘+’, ‘-’, ‘*’, ‘/’ เท่านั้น
  - โอเปอเรเตอร์ (‘*’, ‘/’) มีความสำคัญมากกว่า (‘+’, ‘-’)
- ตัวอย่าง
  - exp = “A+B*C-D/E”
  - postfix = infixToPostfix(exp)
  - print(“Postfix of”, exp, “is”, postfix)
    - โปรแกรมแสดง Postfix of A+B*C-D/E is ABC*+DE/-
- **หมายเหตุ** กำหนดให้สร้าง Stack จากโครงสร้าง Lab 3 และเรียกใช้เมธอดของคลาส ArrayStack เป็นหลัก เช่น push() และ pop() เป็นต้น