```
import random
from typing import List

# A class to represent a Student
class Student:
    def __init__(self, name: str, scores: List[int]):
        self.name = name
        self.scores = scores

    def average_score(self):
        # BUG: Division by len(self.scores) may fail if empty
        return sum(self.scores) / len(self.scores)

    def __str__(self):
        return f"{self.name}: {self.scores}"

# Function to find the top student
def find_top_student(students: List[Student]) -> Student:
    top = students[0]
    for s in students:
        if s.average_score() > top.average_score():
            top = s
    return top

# Function to generate random students
def generate_students(n: int) -> List[Student]:
    names = ["Alice", "Bob", "Charlie", "Diana", "Eve"]
```

```
    students = []
    for i in range(n):
        name = random.choice(names)
        scores = [random.randint(0, 100) for _ in range(random.randint(0, 5))]  # may generate empty list!
        students.append(Student(name, scores))
    return students


# Function with inefficient algorithm (O(n^2))
def find_duplicates(numbers: List[int]) -> List[int]:
    duplicates = []
    for i in range(len(numbers)):
        for j in range(i + 1, len(numbers)):
            if numbers[i] == numbers[j] and numbers[i] not in duplicates:
                duplicates.append(numbers[i])
    return duplicates


if __name__ == "__main__":
    students = generate_students(5)
    for s in students:
        print(s, "Average:", s.average_score())

    top = find_top_student(students)
    print("Top student:", top.name, "with avg", top.average_score())

    nums = [1, 3, 5, 3, 7, 1, 9, 5]
    print("Duplicates:", find_duplicates(nums))
```