Câu 1: Viết 1 prompt yêu cầu **ChatGPT** giúp bạn **tóm tắt kiến thức về Big O Notation** sao cho dễ nhớ để ôn thi.

**(A) Prompt (input)**

Bạn là một trợ giảng lập trình, giúp tôi \*\*tóm tắt kiến thức về Big O Notation\*\* sao cho \*\*ngắn gọn, dễ nhớ\*\* để ôn thi. Yêu cầu:  
1. Giải thích khái niệm Big O trong 1-2 câu.  
2. Liệt kê các lớp độ phức tạp thường gặp (O(1), O(log n), O(n), O(n log n), O(n^2), O(2^n), O(n!)) — mỗi dòng 1 câu ngắn, kèm \*\*1 ví dụ thuật toán\*\* tương ứng.  
3. Cho \*\*1 mẹo/chiến lược ghi nhớ\*\* (mnemonic hoặc trực quan).  
4. Kèm 1 bảng so sánh ngắn (từ nhanh->chậm) và 1 hình ảnh ASCII minh họa nếu được.  
Trả lời bằng tiếng Việt, dùng bullet points.

**(B) Expected output (mẫu)**

* *Khái niệm:* Big O mô tả cách thời gian (hoặc bộ nhớ) tăng theo kích thước đầu vào n, bỏ qua hằng số và bậc thấp hơn.
* *Các lớp phổ biến:*
  + **O(1)** — thời gian không đổi (ví dụ: truy cập phần tử mảng theo chỉ số).
  + **O(log n)** — giảm theo log (ví dụ: tìm kiếm nhị phân).
  + **O(n)** — tuyến tính (ví dụ: duyệt 1 lần để tìm max).
  + **O(n log n)** — thường gặp ở các thuật toán sắp xếp hiệu quả (ví dụ: mergesort, heapsort).
  + **O(n²)** — bình phương (ví dụ: sắp xếp nổi bọt đơn giản, insertion sort tệ nhất).
  + **O(2^n)** — tăng mũ (ví dụ: giải bài toán tập con bằng đệ quy không tối ưu).
  + **O(n!)** — giai thừa (ví dụ: brute-force sắp xếp mọi hoán vị, TSP bằng brute force).
* *Mẹo nhớ:* Sắp xếp các ký hiệu như: 1 < log n < n < n log n < n^2 < 2^n < n!. Hình ảnh trực quan: tưởng tượng n tăng — O(1) là đường thẳng nằm ngang; O(n) là đường thẳng dốc; O(n^2) là đường cong lên rất nhanh; O(2^n)/O(n!) thì nổ tung.
* *Bảng so sánh (nhỏ):* O(1) ≪ O(log n) ≪ O(n) ≪ O(n log n) ≪ O(n²) ≪ O(2^n) ≪ O(n!)
* *ASCII minh họa:*

n: 1 2 3 4 5 6 7   
O(1): \* \* \* \* \* \* \*   
O(n): \* \*\* \*\*\* \*\*\*\* \*\*\*\*\* \*\*\*\*\*\*   
O(n^2): \* \*\* \*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*   
O(2^n): \* \*\* \*\*\*\* \*\*\*\*\*\*\*\* \*\*\*\*\*\* (rất nhanh)

Câu 2: Viết lại prompt trên theo dạng **Iterative Prompting** (hỏi lại để bổ sung ví dụ cụ thể về các thuật toán có độ phức tạp khác nhau).

**(A) Prompt (input)**

Bạn là trợ giảng lập trình. Mục tiêu: giúp tôi \*\*hiểu Big O\*\* và \*\*cung cấp ví dụ thuật toán cụ thể\*\* cho từng độ phức tạp. Hãy làm theo quy trình tương tác (iterative prompting):  
1. Trước tiên, tóm tắt ngắn Big O (2-3 dòng).  
2. Hỏi tôi: "Bạn muốn ví dụ bằng ngôn ngữ nào (Python/Java/C++) và muốn ví dụ nào cho O(n^2) — sắp xếp hay đồ thị?".  
3. Sau khi nhận câu trả lời, đưa \*\*một ví dụ mã\*\* (ngôn ngữ người dùng chọn) cho mỗi lớp độ phức tạp gồm: code ngắn + 1 dòng giải thích cho mỗi ví dụ.  
4. Cuối cùng, cho 1 đề bài ôn tập để tôi tự làm.  
Bắt đầu bằng bước 1 và chờ phản hồi (hãy hỏi câu hỏi ở bước 2).

Câu 3: Viết một prompt **Chain-of-thought** để AI phân tích: *“Những thách thức đạo đức nào có thể phát sinh khi sử dụng AI để tự động hóa hoàn toàn việc kiểm thử phần mềm?”*

**(A) Prompt (input)**

Bạn là chuyên gia đạo đức trong công nghệ và kiểm thử phần mềm. Hãy \*\*phân tích có cấu trúc (structured, step-by-step)\*\* về: "Những thách thức đạo đức có thể phát sinh khi sử dụng AI để tự động hóa hoàn toàn việc kiểm thử phần mềm."   
Yêu cầu:  
1. Liệt kê các vấn đề đạo đức chính (ít nhất 6 mục), mỗi mục kèm 2-3 câu giải thích và 1 ví dụ thực tế.  
2. Đánh giá mức độ rủi ro (cao/trung bình/thấp) cho từng mục và nêu các bên bị ảnh hưởng (ví dụ: người dùng cuối, QA, doanh nghiệp).  
3. Đề xuất \*\*giải pháp giảm thiểu\*\* cho từng vấn đề (kỹ thuật + chính sách).  
4. Tóm tắt 3 lưu ý pháp lý/tuân thủ mà đội phát triển cần kiểm tra trước khi triển khai.  
Trả lời dưới dạng danh sách có tiêu đề các bước: "BƯỚC 1: ...", "BƯỚC 2: ...".  
\*\*Ghi chú:\*\* không yêu cầu mô tả suy nghĩ nội bộ (no private chain-of-thought); chỉ yêu cầu phân tích có cấu trúc.

Câu 4: Viết prompt yêu cầu AI giải thích **nguyên lý hoạt động của Linked List** cho sinh viên mới học.

**(A) Prompt (input)**

Bạn là giảng viên lập trình cơ bản. Giải thích \*\*nguyên lý hoạt động của Linked List\*\* cho sinh viên mới học như sau:  
1. Định nghĩa ngắn (1-2 câu).  
2. Minh họa bằng \*\*sơ đồ ASCII\*\* cho danh sách 4 node.  
3. Mô tả 4 thao tác cơ bản (traversal, insert at head, insert at tail, delete node) kèm \*\*pseudo-code\*\* ngắn cho mỗi thao tác.  
4. Cho 1 ví dụ code Python đơn giản (class Node + class LinkedList + các phương thức trên).  
5. Kèm 3 bài tập nhỏ để sinh viên tự thực hành.  
Trả lời bằng tiếng Việt, ngôn ngữ rõ ràng, không quá học thuật.

Câu 5: Viết 1 prompt chi tiết yêu cầu AI **sinh ví dụ code minh họa một hàm sắp xếp nổi bọt**, sau đó yêu cầu AI giải thích cùng một khái niệm ở **3 cấp độ khác nhau**: cho người mới học, cho sinh viên CNTT, và cho kỹ sư phần mềm.

**(A) Prompt (input)**

Bạn là giảng viên lập trình. Hãy:  
1. Sinh \*\*mã Python\*\* cho hàm `bubble\_sort(arr)` (in-place) có tối ưu: dừng sớm nếu không còn đổi chỗ (early exit).  
2. Sau đó, \*\*giải thích cùng một khái niệm (bubble sort)\*\* ở 3 cấp độ:  
 a) Cho người mới học (tối đa 3 câu, ví dụ bằng thực tế đời thường).   
 b) Cho sinh viên CNTT (mô tả độ phức tạp thời gian, trường hợp tốt/ xấu, khi nào dùng).   
 c) Cho kỹ sư phần mềm (nhấn vào tính ổn định, chi phí, thay thế thực tế và tối ưu).  
3. Kèm 1 ví dụ input + output (arr = [5,2,9,1,5]) và mô tả từng bước (tối đa 6 bước).  
Trả lời bằng tiếng Việt. Đầu ra: code block cho Python + phần giải thích phân đoạn rõ ràng.

Câu 6: Viết 1 prompt yêu cầu AI sinh code một hàm Python để đảo ngược một chuỗi (đầu vào là chuỗi s).

**(A) Prompt (input)**

Viết một hàm Python `reverse\_string(s)` nhận tham số là một chuỗi s và trả về chuỗi đảo ngược. Yêu cầu:  
1. Cho 3 cách hiện thực (kèm comment): slicing, iterative (for loop), và using list reverse.  
2. Mỗi cách kèm 1 dòng giải thích độ phức tạp thời gian và bộ nhớ.  
3. Kèm 2 ví dụ gọi hàm và output.  
Trả lời bằng tiếng Việt, code block Python chạy được.

Câu 7:  
**a) Outline 3 phần (Mở đầu – Nội dung – Kết luận)**

**Outline**

* **MỞ ĐẦU (1 phút)**
  + Giới thiệu bản thân & chủ đề
  + Tại sao AI + Mobile dev quan trọng (một câu dẫn hấp dẫn)
* **NỘI DUNG (8 phút)**
  + Xu hướng hiện tại: AI hỗ trợ design, code, testing, user analytics (2 phút)
  + Case study & demo ngắn (hoặc hình ảnh minh họa) (3 phút)
  + Thách thức & giải pháp (AI bias, privacy, toolchain, CI/CD) (2 phút)
  + Cơ hội nghề nghiệp & kỹ năng cần có (1 phút)
* **KẾT LUẬN (1 phút)**
  + Tóm tắt 3 takeaway chính + kêu gọi hành động (học Flutter/React Native + AI tools)

## **b) Gợi ý 5 slide với nội dung gạch đầu dòng**

**Slide 1 — Tiêu đề + Hook**

* Tên workshop, tên người thuyết trình
* Hook: "AI sẽ rút ngắn thời gian dev từ tuần xuống giờ — nhưng cần kỹ năng mới"

**Slide 2 — AI trong quy trình phát triển mobile**

* Tự động sinh UI từ mô tả (design->code)
* Hỗ trợ viết code (auto-complete, bug suggestion)
* Test tự động, tạo test cases

**Slide 3 — Case study / Demo (hình ảnh)**

* Ví dụ: từ mockup Figma -> mã giao diện (Flutter/React Native)
* Tốc độ giảm bao nhiêu (ví dụ minh hoạ)

**Slide 4 — Thách thức & giải pháp**

* Vấn đề: bias, bảo mật dữ liệu, độ tin cậy test tự động
* Giải pháp: human-in-the-loop, privacy-by-design, validation pipeline

**Slide 5 — Cơ hội nghề nghiệp + Lời kêu gọi hành động**

* Roles: Mobile + ML prompt engineer, MLOps for mobile, mobile AI integration dev
* Học gì: UI/UX, Flutter/React Native, hiểu API AI, testing automation

## **c) Hai ví dụ thực tế về công cụ AI hỗ trợ phát triển app di động**

**Ví dụ 1 — GitHub Copilot / Copilot for VS Code**

* Tự động gợi ý đoạn code, chuyển từ mô tả thành hàm, viết test cơ bản. (Ứng dụng trong mobile: giúp viết logic, xử lý async, state management snippets).

**Ví dụ 2 — FlutterFlow / Figma-to-Code tools**

* Chuyển mockup Figma thành mã Flutter (UI generation), giúp tăng tốc prototyping và handoff.

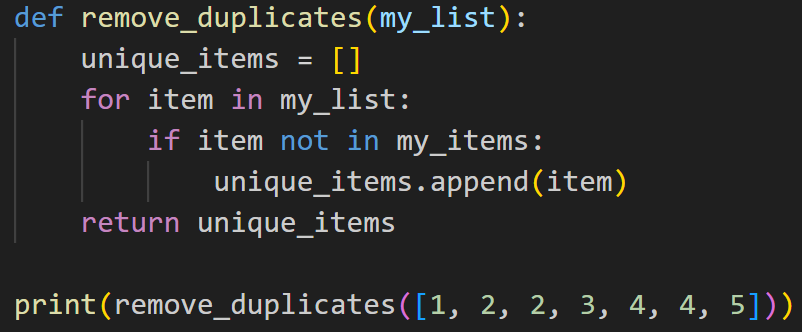
## **d) Script ngắn (~2 phút) cho phần KẾT LUẬN — nhấn mạnh cơ hội nghề nghiệp**

**Script (~2 phút, tiếng Việt):**

"Kết luận lại, AI đang làm thay đổi cách chúng ta xây dựng ứng dụng di động — từ việc tạo giao diện, viết code đến tự động hóa kiểm thử. Điều này mở ra 2 điều quan trọng cho các bạn sinh viên: một là cơ hội nghề nghiệp rất lớn — từ vị trí Mobile Developer truyền thống đến các vai trò mới như 'AI prompt engineer cho app', 'Mobile MLOps', hoặc chuyên gia tích hợp AI vào sản phẩm. Hai là yêu cầu kỹ năng: ngoài ngôn ngữ lập trình như Dart/JavaScript, các bạn cần hiểu cơ bản về mô hình AI, biết cách đánh giá output của AI, và đặc biệt là áp dụng tư duy đạo đức — vì dữ liệu người dùng là trọng yếu.  
 Hãy bắt đầu bằng cách thực hành: lấy một UI mockup, thử chuyển nó sang Flutter bằng tool AI, và tự viết test để kiểm chứng. Nếu các bạn nắm vững bộ kỹ năng này, cơ hội nghề nghiệp sẽ rất rộng mở. Cảm ơn các bạn — mình sẵn sàng trả lời câu hỏi."

Câu 8: Viết 1 prompt yêu cầu AI phân tích và sửa lỗi trong đoạn code sau (được lưu trữ trong file **list\_ops.py**):

**Code ban đầu:**



**(A) Prompt (input)**

Tệp: list\_ops.py (đã có trong project). Yêu cầu:

1. Đọc nội dung file list\_ops.py, phân tích lỗi (cú pháp, logic, edge cases, performance).

2. Sửa lỗi và đưa ra phiên bản file đã sửa (toàn bộ file) với:

- Comment ở trên mỗi hàm giải thích mục đích và thay đổi đã làm.

- Unit tests nhỏ (pytest) minh họa các case chính và edge cases.

3. Nếu sửa đổi thay đổi API hàm, liệt kê breaking changes và cách migrate.

4. Trả lời bằng: (A) tóm tắt lỗi; (B) code sửa (code block); (C) test code; (D) hướng dẫn ngắn chạy test.

Ghi rõ: nếu file không thể chạy do phụ thuộc ngoài, hãy báo rõ dependency và đề xuất mock.

**(B) Output (đầu ra)**

*https://github.com/duong130125/HN-KS23B-NguyenDinhDuong-004.git*

Câu 9:  
**Code mẫu:**  
  
**file: user\_manager.py**

import json

import datetime

from typing import List, Dict, Optional

class userManager: def **init**(self, data\_file: str = "users.json"): self.dataFile = data\_file self.users\_list = [] self.load\_users()

def load\_users(self):  
 try:  
 with open(self.dataFile, 'r', encoding='utf-8') as f:  
 self.users\_list = json.load(f)  
 except FileNotFoundError:  
 self.users\_list = []  
 except json.JSONDecodeError:  
 print("Error decoding JSON file")  
 self.users\_list = []  
  
def save\_users(self):  
 with open(self.dataFile, 'w', encoding='utf-8') as f:  
 json.dump(self.users\_list, f, ensure\_ascii=False, indent=2)  
  
def add\_user(self, userName: str, userEmail: str, userAge: int) -> bool:  
 if not userName or not userEmail:  
 return False  
   
 # Check if user already exists  
 for user in self.users\_list:  
 if user['email'] == userEmail:  
 return False  
   
 new\_user = {  
 'id': len(self.users\_list) + 1,  
 'name': userName,  
 'email': userEmail,  
 'age': userAge,  
 'created\_date': datetime.datetime.now().isoformat(),  
 'is\_active': True  
 }  
   
 self.users\_list.append(new\_user)  
 self.save\_users()  
 return True  
  
def get\_user\_by\_id(self, userId: int) -> Optional[Dict]:  
 for user in self.users\_list:  
 if user['id'] == userId:  
 return user  
 return None  
  
def get\_users\_by\_age\_range(self, min\_age: int, max\_age: int) -> List[Dict]:  
 result\_list = []  
 for user in self.users\_list:  
 if min\_age <= user['age'] <= max\_age:  
 result\_list.append(user)  
 return result\_list  
  
def update\_user\_status(self, userId: int, new\_status: bool) -> bool:  
 user = self.get\_user\_by\_id(userId)  
 if user:  
 user['is\_active'] = new\_status  
 self.save\_users()  
 return True  
 return False  
  
def delete\_user(self, userId: int) -> bool:  
 for i, user in enumerate(self.users\_list):  
 if user['id'] == userId:  
 del self.users\_list[i]  
 self.save\_users()  
 return True  
 return False  
  
def get\_active\_users\_count(self) -> int:  
 count = 0  
 for user in self.users\_list:  
 if user['is\_active']:  
 count += 1  
 return count  
  
def search\_users\_by\_name(self, search\_term: str) -> List[Dict]:  
 results = []  
 for user in self.users\_list:  
 if search\_term.lower() in user['name'].lower():  
 results.append(user)  
 return results  
  
def export\_users\_to\_csv(self, output\_file: str = "users\_export.csv") -> bool:  
 try:  
 import csv  
 with open(output\_file, 'w', newline='', encoding='utf-8') as csvfile:  
 if not self.users\_list:  
 return False  
   
 fieldnames = self.users\_list[0].keys()  
 writer = csv.DictWriter(csvfile, fieldnames=fieldnames)  
 writer.writeheader()  
 for user in self.users\_list:  
 writer.writerow(user)  
 return True  
 except Exception as e:  
 print(f"Export error: {e}")  
 return False

# **Example usage**

if **name** == "**main**": um = userManager("test\_users.json") um.add\_user("Nguyễn Văn An", "[an@example.com](mailto:an@example.com)", 25) um.add\_user("Trần Thị Bình", "[binh@example.com](mailto:binh@example.com)", 30) print(f"Total active users: {um.get\_active\_users\_count()}") print(f"Users aged 20-35: {um.get\_users\_by\_age\_range(20, 35)}")

**PROMPT 1: TẠO TÀI LIỆU HƯỚNG DẪN SỬ DỤNG**

Hãy tạo tài liệu hướng dẫn sử dụng chi tiết cho module userManager trong file user\_manager.py. Tài liệu cần bao gồm:  
  
1. \*\*Mô tả tổng quan\*\*: Giải thích mục đích và chức năng chính của module  
2. \*\*Hướng dẫn cài đặt\*\*: Dependencies và requirements cần thiết  
3. \*\*API Documentation\*\*: Chi tiết về từng method với:  
 - Tham số đầu vào và kiểu dữ liệu  
 - Giá trị trả về  
 - Ví dụ sử dụng cụ thể  
 - Các trường hợp ngoại lệ  
4. \*\*Ví dụ sử dụng thực tế\*\*: Code examples từ cơ bản đến nâng cao  
5. \*\*Best practices\*\*: Các lưu ý khi sử dụng module  
6. \*\*Troubleshooting\*\*: Các lỗi thường gặp và cách khắc phục  
  
Tài liệu nên được viết bằng Markdown format và phù hợp cho developers sử dụng.

## **PROMPT 2: SỬA ĐỔI CODE TOÀN CỤC THEO PEP 8**

Hãy phân tích và sửa đổi toàn bộ code trong file user\_manager.py để tuân thủ 100% tiêu chuẩn PEP 8. Cụ thể:  
  
\*\*PHÂN TÍCH VÀ SỬA CHỮA:\*\*  
1. \*\*Naming conventions\*\*:   
 - Class names: PascalCase  
 - Method names: snake\_case   
 - Variable names: snake\_case  
 - Constants: UPPER\_SNAKE\_CASE  
  
2. \*\*Code formatting\*\*:  
 - Line length (tối đa 79 ký tự)  
 - Indentation (4 spaces)  
 - Blank lines giữa methods và classes  
 - Import statements organization  
 - Whitespace around operators  
  
3. \*\*Documentation\*\*:  
 - Thêm docstrings cho class và methods  
 - Type hints đầy đủ  
 - Comments khi cần thiết  
  
4. \*\*Code structure\*\*:  
 - Tổ chức lại import statements  
 - Cải thiện error handling  
 - Optimize performance where possible  
  
\*\*YÊU CẦU:\*\*  
- Giữ nguyên functionality  
- Đưa ra báo cáo chi tiết về những thay đổi đã thực hiện  
- Giải thích lý do cho mỗi thay đổi  
- Đảm bảo code vẫn hoạt động như cũ  
  
Hãy rewrite toàn bộ file với code đã được cải thiện.

**(B) Output (đầu ra)**

*https://github.com/duong130125/HN-KS23B-NguyenDinhDuong-004.git*