ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

BÁO CÁO BÀI TẬP

Kỹ năng viết prompt hiệu quả cho mô hình ngôn ngữ lớn

Sinh viên: Chu Nhật Anh

Lớp: RE2

Khoa: Điện tử viễn thông

I. MỤC TIÊU

Bài tập này dùng để luyện cách viết prompt sao cho hỏi AI hiệu quả hơn. Cụ thể là chọn 3 việc hay làm trong học tập, rồi thử viết 3 kiểu prompt khác nhau (từ đơn giản đến phức tạp hơn) và xem kết quả ra sao.

II. 3 TÁC VỤ HỌC TẬP ĐƯỢC CHỌN

3 tác vụ hay dùng nhất khi học:

- Tóm tắt bài đọc/tài liệu học thuật
- Giải thích khái niệm khó (ở đây em chọn đệ quy trong lập trình)
- Tạo câu hỏi ôn tập môn Mạng máy tính

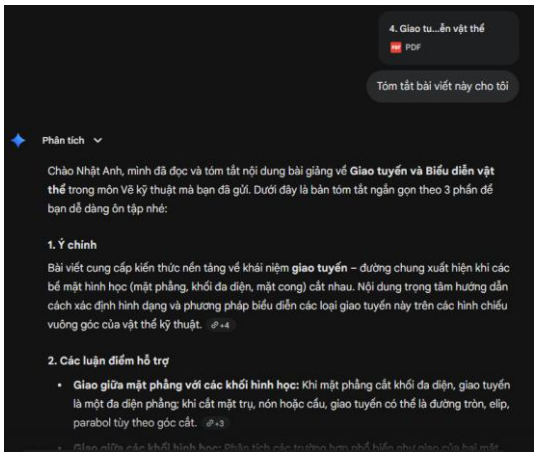
III. CÁC PHIÊN BẢN PROMPT VÀ KẾT QUẢ THỬ NGHIỆM

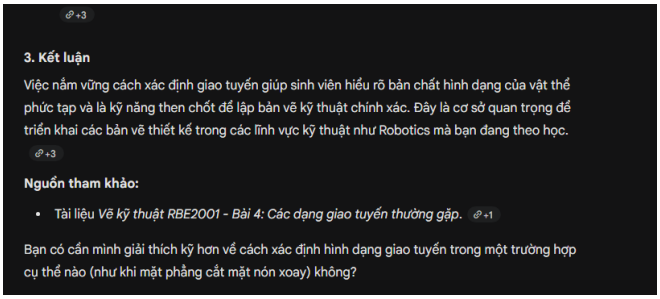
1. Tác vụ: Tóm tắt tài liệu học thuật

Prompt cơ bản:

> "Tóm tắt bài viết này cho tôi."

Kết quả: AI tóm tắt được nhưng khá sơ sài, có ý chính, ý phụ nhưng chưa rõ ràng. Câu trả lời cũng hơi dài dòng.

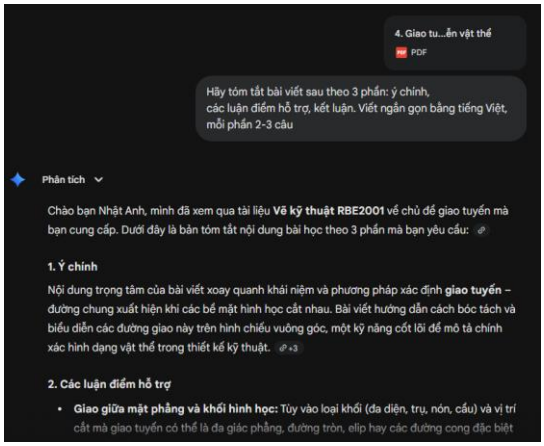


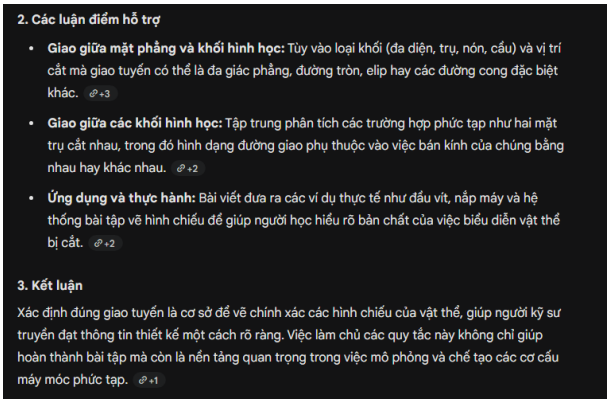


Prompt cải tiến:

> "Hãy tóm tắt bài viết sau theo 3 phần: ý chính, các luận điểm hỗ trợ, kết luận. Viết ngắn gọn bằng tiếng Việt, mỗi phần 2-3 câu."

Kết quả: Tốt hơn rõ rệt, có bố cục rõ ràng, dễ đọc hơn. Vẫn còn đôi chỗ hơi chung chung, không quá khác biệt so với dung prompt cơ bản.

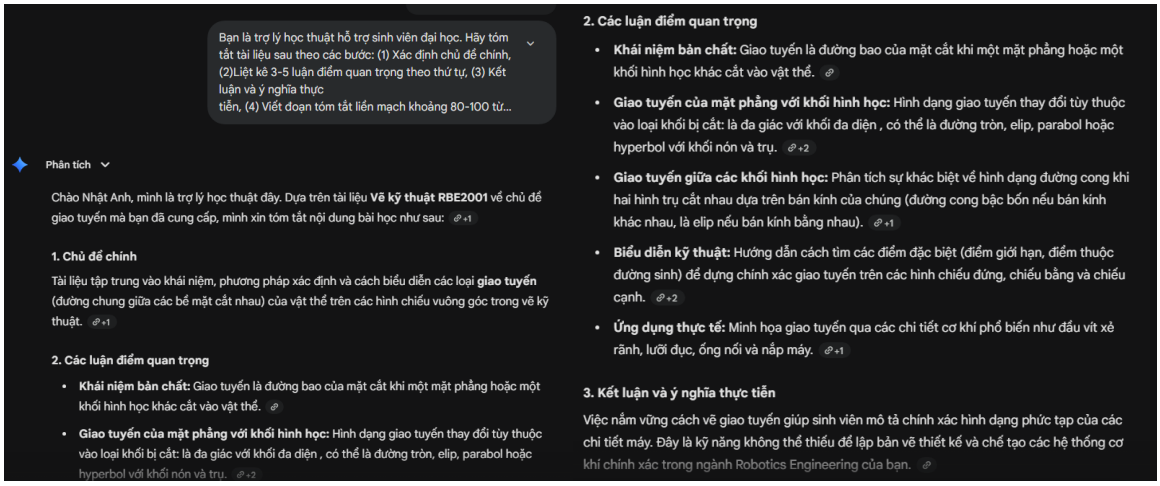


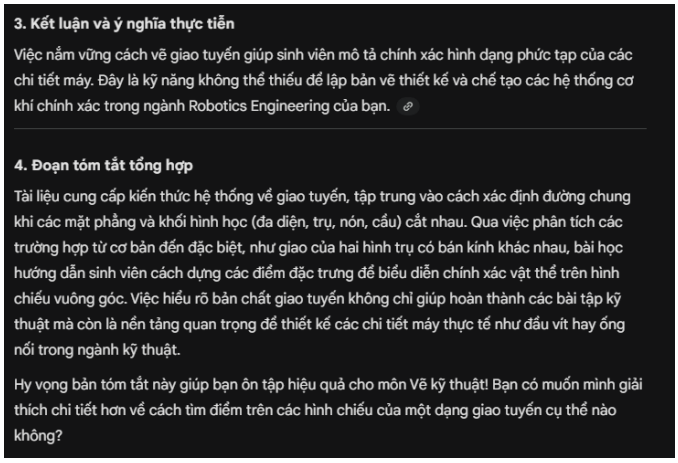


Prompt nâng cao:

> "Bạn là trợ lý học thuật hỗ trợ sinh viên đại học. Hãy tóm tắt tài liệu sau theo các bước: (1) Xác định chủ đề chính, (2) Liệt kê 3-5 luận điểm quan trọng theo thứ tự, (3) Kết luận và ý nghĩa thực tiễn, (4) Viết đoạn tóm tắt liền mạch khoảng 80-100 từ bằng tiếng Việt. Tài liệu: [dán nội dung vào đây]"

Kết quả: Rất tốt, đầu ra có cấu trúc rõ, đủ ý, dùng được luôn không cần sửa nhiều.



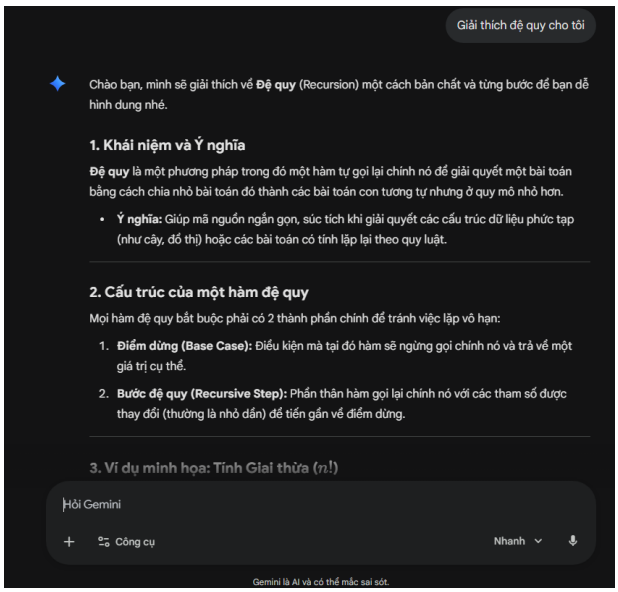


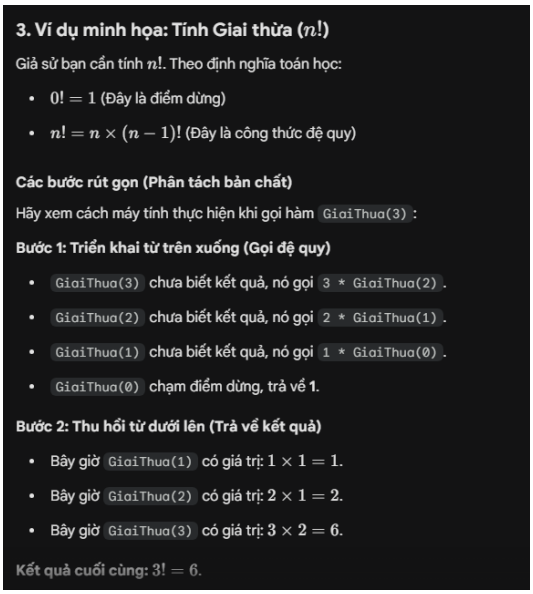
2. Tác vụ: Giải thích khái niệm đệ quy

Prompt cơ bản:

"Giải thích đệ quy cho tôi."

Kết quả: Giải thích đúng nhưng hơi khô và nặng về lý thuyết, vẫn có ví dụ nhưng không đáng kể, không có code ví dụ, khó hiểu với người mới.

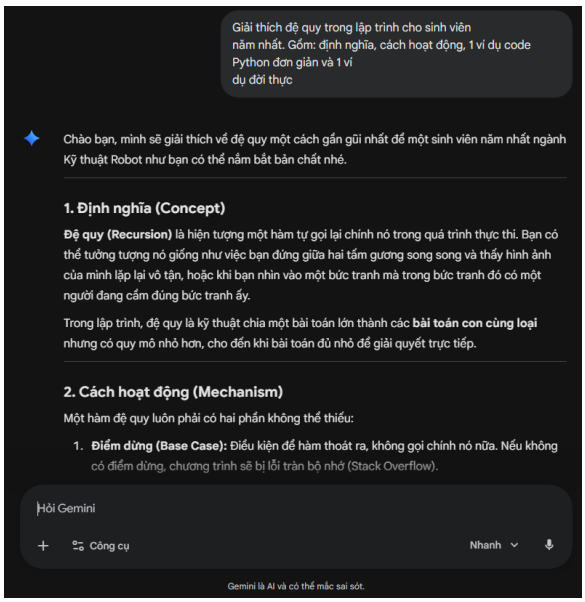


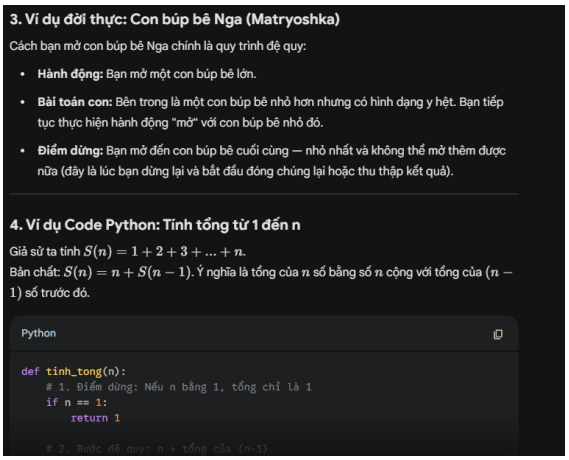


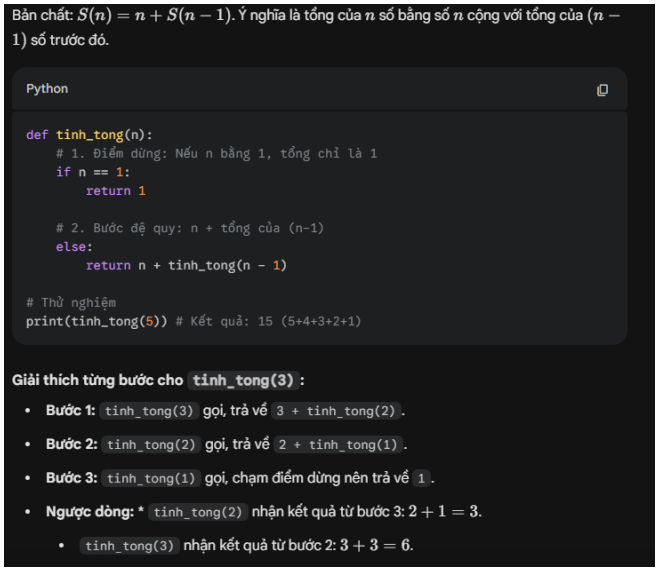
Prompt cải tiến:

"Giải thích đệ quy trong lập trình cho sinh viên năm nhất. Gồm: định nghĩa, cách hoạt động, 1 ví dụ code Python đơn giản và 1 ví dụ đời thực."

Kết quả: Dễ hiểu hơn nhiều, có ví dụ thực tế gần gũi. Có code Python.







Bản chất: $S(n) = n + S(n-1)$. Ý nghĩa là tổng của n số bằng số n cộng với tổng của $(n-1)$ số trước đó.

Python

```
def tinh_tong(n):
    # 1. Điểm dừng: Nếu n bằng 1, tổng chỉ là 1
    if n == 1:
        return 1

    # 2. Bước đệ quy: n + tổng của (n-1)
    else:
        return n + tinh_tong(n - 1)

# Thử nghiệm
print(tinh_tong(5)) # Kết quả: 15 (5+4+3+2+1)
```

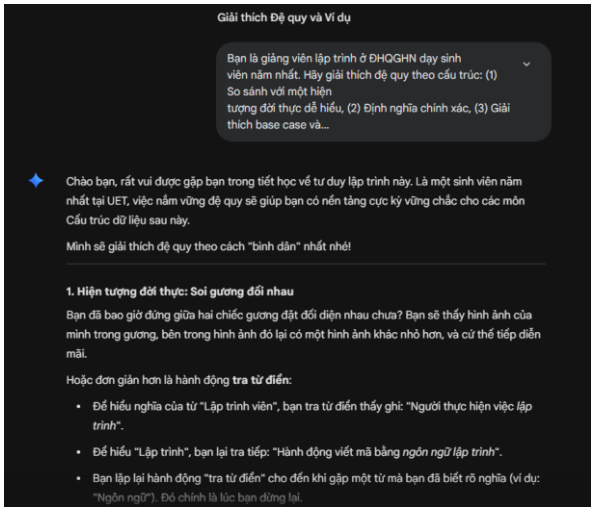
Giải thích từng bước cho `tinh_tong(3)` :

- **Bước 1:** `tinh_tong(3)` gọi, trả về `3 + tinh_tong(2)` .
- **Bước 2:** `tinh_tong(2)` gọi, trả về `2 + tinh_tong(1)` .
- **Bước 3:** `tinh_tong(1)` gọi, chạm điểm dừng nên trả về `1` .
- **Ngược dòng:** * `tinh_tong(2)` nhận kết quả từ bước 3: $2 + 1 = 3$.
  - `tinh_tong(3)` nhận kết quả từ bước 2: $3 + 3 = 6$.

Prompt nâng cao:

> "Bạn là giảng viên lập trình ở ĐHQGHN dạy sinh viên năm nhất. Hãy giải thích đệ quy theo cấu trúc: (1) So sánh với một hiện tượng đời thực dễ hiểu, (2) Định nghĩa chính xác, (3) Giải thích base case và recursive case, (4) Code Python ví dụ có chú thích từng dòng, (5) Lỗi hay gặp (vòng lặp vô hạn), (6) 1 câu hỏi nhỏ để tự kiểm tra. Phong cách: đơn giản, dễ hiểu, không dùng thuật ngữ phức tạp."

Kết quả: Giải thích rất dễ hiểu, có analogy hay, code có chú thích rõ, cuối có câu hỏi để tự kiểm tra. Đây là phiên bản tốt nhất.



Giải thích Đệ quy và Ví dụ

Bạn là giảng viên lập trình ở ĐHQGHN dạy sinh viên năm nhất. Hãy giải thích đệ quy theo cấu trúc: (1) So sánh với một hiện tượng đời thực dễ hiểu, (2) Định nghĩa chính xác, (3) Giải thích base case và...

Chào bạn, rất vui được gặp bạn trong tiết học về tư duy lập trình này. Là một sinh viên năm nhất tại UET, việc nắm vững đệ quy sẽ giúp bạn có nền tảng cực kỳ vững chắc cho các môn Cấu trúc dữ liệu sau này.

Mình sẽ giải thích đệ quy theo cách "bình dân" nhất nhé!

1. Hiện tượng đời thực: Soi gương đối nhau

Bạn đã bao giờ đứng giữa hai chiếc gương đặt đối diện nhau chưa? Bạn sẽ thấy hình ảnh của mình trong gương, bên trong hình ảnh đó lại có một hình ảnh khác nhỏ hơn, và cứ thế tiếp diễn mãi.

Hoặc đơn giản hơn là hành động **tra từ điển**:

- Để hiểu nghĩa của từ "Lập trình viên", bạn tra từ điển thấy ghi: "Người thực hiện việc *lập trình*".
- Để hiểu "Lập trình", bạn lại tra tiếp: "Hành động viết mã bằng *ngôn ngữ lập trình*".
- Bạn lặp lại hành động "tra từ điển" cho đến khi gặp một từ mà bạn đã biết rõ nghĩa (ví dụ: "Ngôn ngữ"). Đó chính là lúc bạn dừng lại.



2. Định nghĩa chính xác

Đệ quy là một phương pháp trong lập trình mà trong đó **một hàm tự gọi lại chính nó** để giải quyết một bài toán lớn bằng cách chia nhỏ nó thành các bài toán con tương tự.

3. Hai thành phần bắt buộc: Base case và Recursive case

Để một hàm đệ quy hoạt động mà không làm treo máy, nó luôn cần 2 phần:

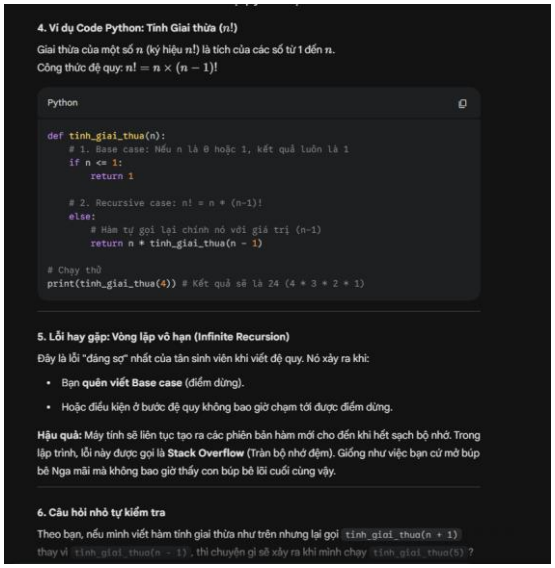
- **Base case (Điểm dừng):** Đây là trường hợp đơn giản nhất mà chúng ta đã biết sẵn câu trả lời, không cần phải tính toán thêm. Nó giúp hàm thoát ra, không gọi chính nó nữa.
- **Recursive case (Bước đệ quy):** Đây là lúc hàm tự gọi lại chính mình nhưng với một giá trị nhỏ hơn hoặc đơn giản hơn, nhằm tiến dần về điểm dừng.

4. Ví dụ Code Python: Tính Giai thừa ($n!$)

Giai thừa của một số n (ký hiệu $n!$) là tích của các số từ 1 đến n.
Công thức đệ quy: $n! = n \times (n-1)!$

Python

```
def tinh_giai_thua(n):
    # 1. Base case: Nếu n là 0 hoặc 1, kết quả luôn là 1
    if n <= 1:
        return 1
```



4. Ví dụ Code Python: Tính Giai thừa (n!)

Giai thừa của một số n (ký hiệu $n!$) là tích của các số từ 1 đến n.
Công thức đệ quy: $n! = n \times (n-1)!$

Python

```
def tinh_giai_thua(n):
    # 1. Base case: Nếu n là 0 hoặc 1, kết quả luôn là 1
    if n <= 1:
        return 1

    # 2. Recursive case: n! = n * (n-1)!
    else:
        # Hàm tự gọi lại chính nó với giá trị (n-1)
        return n * tinh_giai_thua(n - 1)

# Chạy thử
print(tinh_giai_thua(4)) # Kết quả sẽ là 24 (4 * 3 * 2 * 1)
```

5. Lỗi hay gặp: Vòng lặp vô hạn (Infinite Recursion)

Đây là lỗi "đáng sợ" nhất của tân sinh viên khi viết đệ quy. Nó xảy ra khi:

- Bạn **quên viết Base case** (điểm dừng).
- Hoặc điều kiện ở bước đệ quy không bao giờ chạm tới được điểm dừng.

Hậu quả: Máy tính sẽ liên tục tạo ra các phiên bản hàm mới cho đến khi hết sạch bộ nhớ. Trong lập trình, lỗi này được gọi là **Stack Overflow** (Tràn bộ nhớ đệm). Giống như việc bạn cứ mở búp bê Nga mãi mà không bao giờ thấy con búp bê lõi cuối cùng vậy.

6. Câu hỏi nhỏ tự kiểm tra

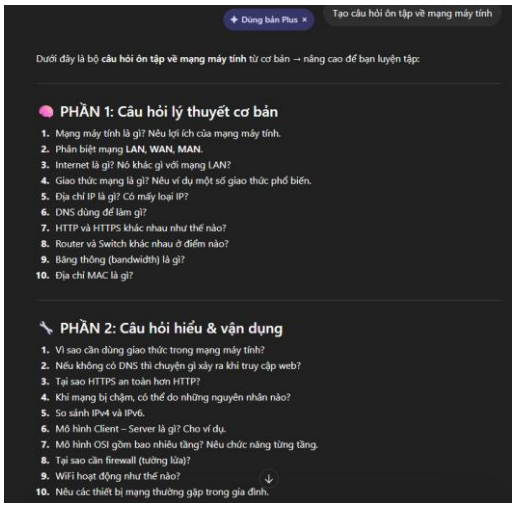
Theo bạn, nếu mình viết hàm tính giai thừa như trên nhưng lại gọi `tinh_giai_thua(n + 1)` thay vì `tinh_giai_thua(n - 1)`, thì chuyện gì sẽ xảy ra khi mình chạy `tinh_giai_thua(5)` ?

3. Tác vụ: Tạo câu hỏi ôn tập Mạng máy tính

Prompt cơ bản:

"Tạo câu hỏi ôn tập về mạng máy tính."

Kết quả: Câu hỏi xuất hiện nhưng chủ yếu là trắc nghiệm đơn giản, không có đáp án, các chủ đề không đều nhau.



Dùng bản Plus ×

Tạo câu hỏi ôn tập về mạng máy tính

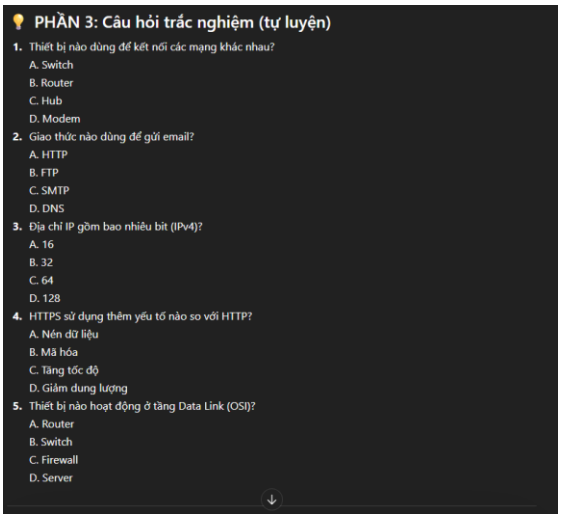
Dưới đây là bộ **câu hỏi ôn tập về mạng máy tính** từ cơ bản → nâng cao để bạn luyện tập:

PHẦN 1: Câu hỏi lý thuyết cơ bản

1. Mạng máy tính là gì? Nêu lợi ích của mạng máy tính.
2. Phân biệt mạng **LAN**, **WAN**, **MAN**.
3. Internet là gì? Nó khác gì với mạng LAN?
4. Giao thức mạng là gì? Nêu ví dụ một số giao thức phổ biến.
5. Địa chỉ IP là gì? Có mấy loại IP?
6. DNS dùng để làm gì?
7. HTTP và HTTPS khác nhau như thế nào?
8. Router và Switch khác nhau ở điểm nào?
9. Băng thông (bandwidth) là gì?
10. Địa chỉ MAC là gì?

PHẦN 2: Câu hỏi hiểu & vận dụng

1. Vì sao cần dùng giao thức trong mạng máy tính?
2. Nếu không có DNS thì chuyện gì xảy ra khi truy cập web?
3. Tại sao HTTPS an toàn hơn HTTP?
4. Khi mạng bị chậm, có thể do những nguyên nhân nào?
5. So sánh IPv4 và IPv6.
6. Mô hình Client – Server là gì? Cho ví dụ.
7. Mô hình OSI gồm bao nhiêu tầng? Nêu chức năng từng tầng.
8. Tại sao cần firewall (tường lửa)?
9. WiFi hoạt động như thế nào?
10. Nêu các thiết bị mạng thường gặp trong gia đình.



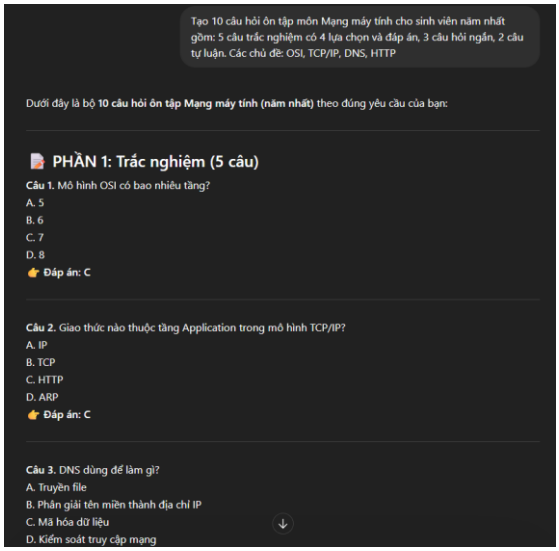
PHẦN 3: Câu hỏi trắc nghiệm (tự luyện)

1. Thiết bị nào dùng để kết nối các mạng khác nhau?
   A. Switch
   B. Router
   C. Hub
   D. Modem
2. Giao thức nào dùng để gửi email?
   A. HTTP
   B. FTP
   C. SMTP
   D. DNS
3. Địa chỉ IP gồm bao nhiêu bit (IPv4)?
   A. 16
   B. 32
   C. 64
   D. 128
4. HTTPS sử dụng thêm yếu tố nào so với HTTP?
   A. Nén dữ liệu
   B. Mã hóa
   C. Tăng tốc độ
   D. Giảm dung lượng
5. Thiết bị nào hoạt động ở tầng Data Link (OSI)?
   A. Router
   B. Switch
   C. Firewall
   D. Server

Prompt cải tiến:

"Tạo 10 câu hỏi ôn tập môn Mạng máy tính cho sinh viên năm nhất gồm: 5 câu trắc nghiệm có 4 lựa chọn và đáp án, 3 câu hỏi ngắn, 2 câu tự luận. Các chủ đề: OSI, TCP/IP, DNS, HTTP."

Kết quả: Đủ số lượng, chỉ có đáp án phần trắc nghiệm, bao quát các chủ đề yêu cầu. Tuy nhiên câu tự luận còn hơi đơn giản.



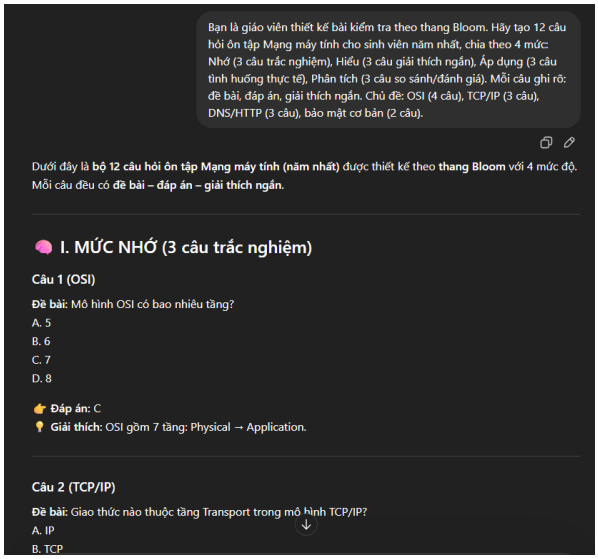




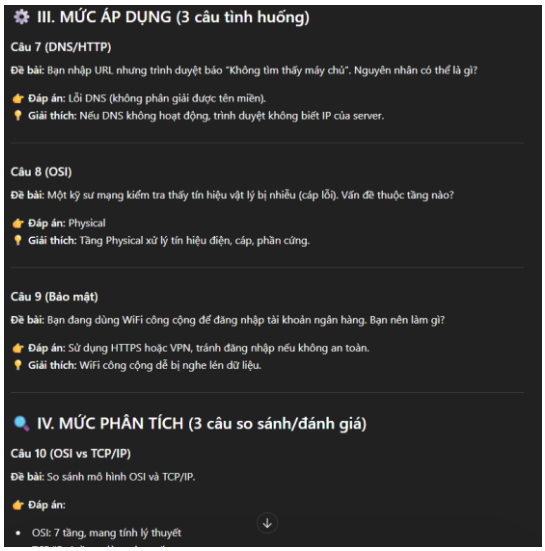
Prompt nâng cao:

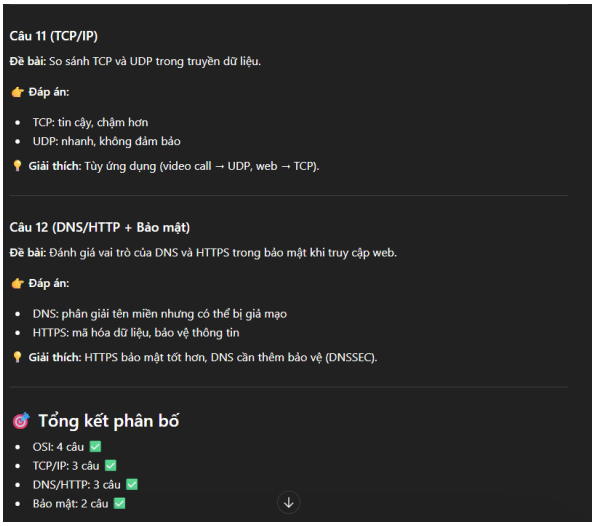
> "Bạn là giáo viên thiết kế bài kiểm tra theo thang Bloom. Hãy tạo 12 câu hỏi ôn tập Mạng máy tính cho sinh viên năm nhất, chia theo 4 mức: Nhớ (3 câu trắc nghiệm), Hiểu (3 câu giải thích ngắn), Áp dụng (3 câu tình huống thực tế), Phân tích (3 câu so sánh/đánh giá). Mỗi câu ghi rõ: đề bài, đáp án, giải thích ngắn. Chủ đề: OSI (4 câu), TCP/IP (3 câu), DNS/HTTP (3 câu), bảo mật cơ bản (2 câu)."

Kết quả: Bộ câu hỏi rất đầy đủ, có phân cấp độ khó rõ, mỗi câu đều có giải thích. Dùng ôn tập rất tiện.









IV. BẢNG SO SÁNH KẾT QUẢ

Tác vụ 1 – Tóm tắt tài liệu:

Tiêu chí đánh giá	Prompt cơ bản	Prompt cải tiến	Prompt nâng cao
Có cấu trúc rõ không?	Không	Có	Có, rất rõ
Đủ ý chính không?	Thiếu	Đủ	Đầy đủ
Dùng ngay được không?	Phải sửa nhiều	Sửa ít	Gần như dùng được ngay
Điểm (1-10)	4	7	9

Tác vụ 2 – Giải thích khái niệm:

Tiêu chí đánh giá	Prompt cơ bản	Prompt cải tiến	Prompt nâng cao
Dễ hiểu với SV năm nhất?	Khó hiểu	Tạm được	Rất dễ hiểu
Có ví dụ thực tế không?	Không	Có 1	Có nhiều
Code có chú thích không?	Không có code	Có	Có đầy đủ
Điểm (1-10)	3	7	9.5

Tác vụ 3 – Câu hỏi ôn tập:

Tiêu chí đánh giá	Prompt cơ bản	Prompt cải tiến	Prompt nâng cao
Đủ số lượng yêu cầu?	Không ổn định	Đúng 10 câu	Đúng 12 câu
Đa dạng hình thức?	Chỉ trắc nghiệm	3 dạng	4 cấp độ Bloom
Có đáp án + giải thích?	Không	Có đáp án 1 phần	Có cả hai
Điểm (1-10)	3.5	7	9.5

V. PHÂN TÍCH – TẠI SAO PROMPT NÂNG CAO HIỆU QUẢ HƠN?

Qua thử nghiệm, em nhận ra có 4 lý do chính:

1. Đặt vai trò cho AI (Role Prompting)

Khi em nói "bạn là giảng viên dạy sinh viên năm nhất", AI chọn cách giải thích đơn giản, dùng ví dụ gần gũi hơn. Còn khi không nói gì, AI giải thích theo kiểu chung chung, không biết đối tượng là ai nên dễ bị quá khó hoặc quá dễ.

2. Chỉ định rõ cấu trúc đầu ra

Prompt cơ bản không nói muốn format gì nên AI tự chọn, kết quả không đồng đều. Khi mình nói rõ "3 phần", "5 câu trắc nghiệm + 3 câu ngắn"... thì AI làm đúng theo, không cần sửa.

3. Yêu cầu làm từng bước (Chain-of-Thought)

Với tác vụ tóm tắt, khi em yêu cầu AI làm từng bước (xác định chủ đề → liệt kê luận điểm → viết tóm tắt), kết quả logic hơn, ít bỏ sót ý hơn so với chỉ nói "tóm tắt cho tôi".

4. Cho ví dụ mẫu (Few-shot)

Ở tác vụ giải thích đệ quy, em đưa ví dụ về cách giải thích vòng lặp theo kiểu analogy trước. AI hiểu ngay em muốn phong cách nào và làm theo, kết quả giống ý mình hơn nhiều so với chỉ mô tả bằng lời.

VI. TỔNG HỢP – MẸO VIẾT PROMPT HIỆU QUẢ

Sau khi thử nghiệm, em rút ra được mấy điểm sau:

Mẹo	Cách làm cụ thể	Khi nào dùng
Đặt vai trò	Bắt đầu bằng "Bạn là [vai trò]..."	Giải thích, hướng dẫn
Chỉ định format	Nói rõ số lượng, cấu trúc cần có	Báo cáo, câu hỏi, danh sách
Làm từng bước	Liệt kê các bước xử lý cụ thể	Tóm tắt, phân tích
Cho ví dụ mẫu	Cung cấp 1 ví dụ ngắn về output mong muốn	Khi khó diễn đạt bằng lời
Gắn bối cảnh	Nói rõ đối tượng đọc và mục đích	Giải thích cho người cụ thể

Nguyên tắc quan trọng nhất: prompt càng cụ thể thì kết quả càng tốt. Không cần phải viết dài, chỉ cần nói rõ mình muốn gì, muốn theo format nào và muốn AI đóng vai gì.

VII. KẾT LUẬN

Qua bài tập này, em thấy kỹ năng viết prompt thực sự quan trọng. Cùng một câu hỏi nhưng cách hỏi khác nhau thì kết quả chênh nhau rất nhiều (từ 3-4 điểm lên 9-9.5 điểm). Prompt nâng cao không nhất thiết phải dài, mà cần đúng – đủ vai trò, đủ cấu trúc, đủ bối cảnh.

Em sẽ tiếp tục áp dụng các kỹ thuật này khi dùng AI để học các môn khác, đặc biệt là khi cần giải thích bài tập lập trình và ôn thi.