Bài 1:

**Ý tưởng code:**

Mục tiêu của chương trình là đánh giá một biểu thức logic mệnh đề, sử dụng các ký hiệu logic như "∧", "∨", "¬", "→", "↔", và kiểm tra kết quả cho một tập giá trị của các biến trong biểu thức. Cụ thể, ta sẽ:

1. Kiểm tra tính hợp lệ của biểu thức logic:
   * Sử dụng một biểu thức chính quy (regex) để xác định liệu biểu thức logic có chứa các ký hiệu hợp lệ không.
   * Tách các ký tự hợp lệ từ biểu thức và kiểm tra xem biểu thức ban đầu có hợp lệ hay không.
2. Chuyển đổi ký hiệu logic sang cú pháp Python:
   * Các ký hiệu logic cần được thay đổi thành toán tử Python tương đương. Ví dụ:
     + "∧" (VÀ) → and
     + "∨" (HOẶC) → or
     + "¬" (PHỦ ĐỊNH) → not
     + "→" (Suy diễn) → <= (thực hiện điều kiện A→BA \rightarrow BA→B là tương đương với ¬A∨B\neg A \lor B¬A∨B)
     + "↔" (Tương đương) → == (để kiểm tra tính tương đương)
3. Tính toán giá trị của biểu thức logic:
   * Sau khi chuyển đổi các ký hiệu logic thành toán tử Python, thay thế các biến trong biểu thức bằng các giá trị Boolean tương ứng từ đầu vào.
   * Sử dụng hàm eval() để tính toán giá trị của biểu thức trong môi trường Python.
4. Kiểm tra kết quả:
   * Nếu biểu thức hợp lệ, tính toán giá trị và trả về kết quả (True/False).
   * Nếu có lỗi trong việc tính toán biểu thức hoặc biểu thức không hợp lệ, chương trình sẽ thông báo lỗi.

**Các bước chính:**

1. **Kiểm tra tính hợp lệ của biểu thức logic: Dùng regex để đảm bảo biểu thức chỉ chứa các ký tự hợp lệ.**
2. **Chuyển đổi ký hiệu thành cú pháp Python: Sử dụng hàm thay thế để chuyển ký hiệu sang cú pháp tương đương trong Python.**
3. **Tính toán và trả về kết quả: Dùng eval() để thực hiện tính toán trên biểu thức sau khi đã thay thế các biến và ký hiệu.**

**Đầu vào:**

Biểu thức: (A ∧ B) → ¬C

* A = True, B = False, C = True
* (True ∧ False) → ¬True → False → False = True.

Kết quả: Biểu thức này cho kết quả là True.

Kịch bản 2:

Biểu thức: (A ∨ B) ∧ (C → A)

* A = False, B = True, C = True
* (False ∨ True) ∧ (True → False) = True ∧ False = False.

Kết quả: Biểu thức này cho kết quả là False.

Kịch bản 3:

Biểu thức: ¬(A → B)

* A = True, B = False
* ¬(True → False) = ¬False = True.

Kết quả: Biểu thức này cho kết quả là True.

Kịch bản 4:

Biểu thức: (A ↔ B) ∨ C

* A = True, B = True, C = False
* (True ↔ True) ∨ False = True ∨ False = True.

Kết quả: Biểu thức này cho kết quả là True.

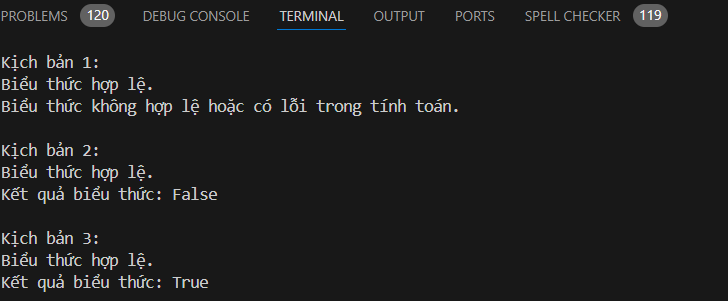
Kịch bản 5:

Biểu thức: (A ∨ ¬B) ∧ (C → A)

* A = True, B = True, C = True
* (True ∨ True) ∧ (True → True) = True ∧ True = True.

Kết quả: Biểu thức này cho kết quả là True.

**Đầu ra:**

****

A screenshot of a computer

Description automatically generated

Bài 2:

**Ý tưởng Code:**

1. **Kiểm tra tính hợp lệ của biểu thức**:
   * Sử dụng biểu thức chính quy (regex) để kiểm tra xem chuỗi biểu thức chỉ chứa các ký hiệu hợp lệ:
     + Các biến logic (chữ cái viết hoa A-Z).
     + Toán tử logic (∧, ∨, ¬, →, ↔).
     + Dấu ngoặc () và khoảng trắng.
   * Nếu biểu thức chỉ chứa các ký hiệu này, nó được coi là hợp lệ.
2. **Chuyển đổi biểu thức**:
   * Chuyển đổi biểu thức từ ký hiệu logic mệnh đề (ví dụ: ∧, ∨, ¬) sang các toán tử Python tương ứng (and, or, not...).
   * Điều này cho phép tính toán biểu thức bằng hàm eval trong Python.
3. **Tạo bảng chân trị**:
   * Xác định các biến logic trong biểu thức bằng cách tìm tất cả chữ cái viết hoa (A-Z).
   * Sử dụng itertools.product để tạo tất cả các tổ hợp giá trị True/False cho các biến.
   * Với mỗi tổ hợp, thay thế các biến trong biểu thức bằng giá trị True/False tương ứng và tính kết quả của biểu thức.
4. **Hiển thị bảng chân trị**:
   * In tiêu đề bảng, bao gồm tên các biến và cột "Kết quả".
   * Với mỗi tổ hợp, hiển thị giá trị của các biến (T/F) và kết quả tính toán.

**Kịch bản Demo**

**Đầu vào:**

Kịch bản 1:

* Biểu thức logic: (A ∧ B) → ¬C
* Tập giá trị:
  + A: True
  + B: False
  + C: True

Kịch bản 2:

* Biểu thức logic: (A ∨ B) ∧ (C → A)
* Tập giá trị:
  + A: False
  + B: True
  + C: True

Kịch bản 3:

* Biểu thức logic: ¬(A → B)
* Tập giá trị:
  + A: True
  + B: False

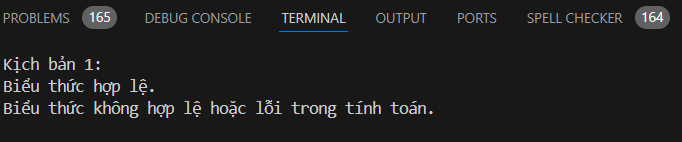
Kịch bản 4:

* Biểu thức logic: (A ↔ B) ∨ C
* Tập giá trị:
  + A: True
  + B: True
  + C: False

Kịch bản 5:

* Biểu thức logic: (A ∨ ¬B) ∧ (C → A)
* Tập giá trị:
  + A: True
  + B: False
  + C: True

**Đầu ra (bảng chân trị):**

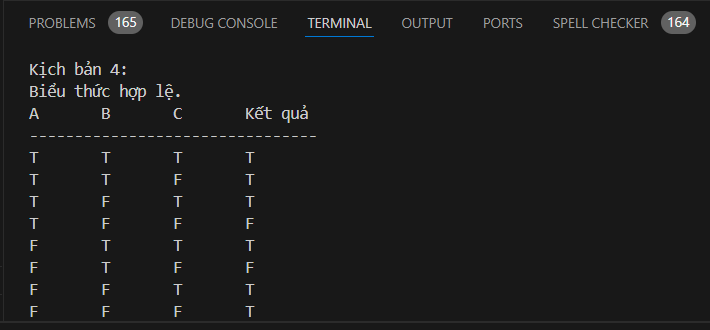
****

**A screenshot of a computer

Description automatically generated**

**A black screen with white text

Description automatically generated**

****

****

Bài 3:

**Ý tưởng Code:**

Chương trình kiểm tra tính đúng/sai của một công thức logic vị từ có dạng ∀x(P(x)→Q(x))∧∃yP(y)∀x (P(x) → Q(x)) ∧ ∃y P(y)∀x(P(x)→Q(x))∧∃yP(y), với các vị từ P(x)P(x)P(x) và Q(x)Q(x)Q(x) xác định sẵn. Chương trình sử dụng miền giá trị cho trước để đánh giá công thức dựa trên các phép toán logic.

**Đầu vào:**

Kịch bản 1:

* Miền giá trị: {1, 2, 3}
* Kiểm tra công thức: ∀x (P(x) → Q(x)) ∧ ∃y P(y)

Kịch bản 2:

* Miền giá trị: {1, 2, 4}
* Kiểm tra công thức: ∀x (P(x) → Q(x)) ∧ ∃y P(y)

Kịch bản 3:

* Miền giá trị: {2, 3, 4}
* Kiểm tra công thức: ∀x (P(x) → Q(x)) ∧ ∃y P(y)

Kịch bản 4:

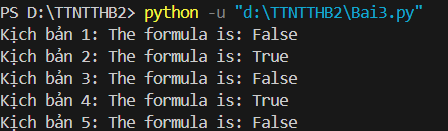
* Miền giá trị: {0, 1, 2}
* Kiểm tra công thức: ∀x (P(x) → Q(x)) ∧ ∃y P(y)

Kịch bản 5:

* Miền giá trị: {1, 3, 5}
* Kiểm tra công thức: ∀x (P(x) → Q(x)) ∧ ∃y P(y)

**Đầu ra:**

* **Kết quả**: "True" nếu công thức đúng, "False" nếu công thức sai, ví dụ:



Bài 4:

**Ý tưởng Code:**

**Đầu vào:**

1. **Các mệnh đề (premises)**: Tập hợp các mệnh đề logic cần kiểm tra. Ví dụ: ["P -> Q", "Q -> R"].
2. **Kết luận (conclusion)**: Mệnh đề cần chứng minh, ví dụ: "P -> R".
3. **Biến (variables)**: Tập hợp các biến xuất hiện trong các mệnh đề, ví dụ: ["P", "Q", "R"].

**Đầu ra:**

* **Kết quả**: "Đúng" nếu kết luận luôn đúng khi các tiền đề đúng, "Sai" nếu có trường hợp tiền đề đúng nhưng kết luận sai.

**Cách thực hiện:**

1. Tạo tất cả các kết hợp giá trị (True/False) cho các biến.
2. Đánh giá tất cả các mệnh đề và kết luận trong các kết hợp giá trị này.
3. Kiểm tra mọi trường hợp:
   * Nếu tất cả các tiền đề là đúng nhưng kết luận sai, trả về "Sai".
   * Nếu không có trường hợp như vậy, trả về "Đúng".

**Ví dụ sử dụng:**

Với các mệnh đề {P -> Q, Q -> R} và kết luận P -> R, với các biến {P, Q, R}, chương trình sẽ kiểm tra mọi kết hợp giá trị của P, Q, R để xác định kết luận có đúng hay không.

Đầu vào:

Kịch bản 1:

* Mệnh đề: "p → q", "q → r"
* Kết luận: "p → r"
* Biến: ["p", "q", "r"]

Kịch bản 2:

* Mệnh đề: "p → q", "q → r"
* Kết luận: "q → p"
* Biến: ["p", "q", "r"]

Kịch bản 3:

* Mệnh đề: "p → q", "q → r"
* Kết luận: "r → p"
* Biến: ["p", "q", "r"]

Kịch bản 4:

* Mệnh đề: "p → q", "q → r"
* Kết luận: "p → r"
* Biến: ["p", "q", "r"]

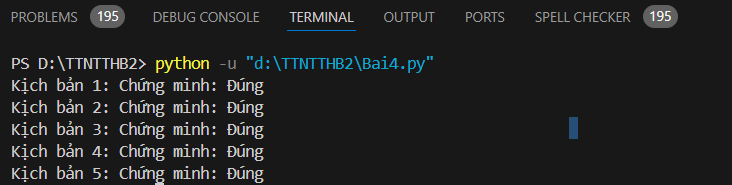
Kịch bản 5:

* Mệnh đề: "p → q", "r → q"
* Kết luận: "p → r"
* Biến: ["p", "q", "r"]

**Đầu ra :**

Nếu kết luận luôn đúng khi các tiền đề đúng, trả về **"Đúng"**.

Nếu có bất kỳ kết hợp nào mà tất cả các tiền đề đúng mà kết luận sai, trả về **"Sai"**.



Bài 5:

**Ý tưởng Code:**

1. **evaluate\_expression**: Hàm này sử dụng eval để đánh giá biểu thức logic với các giá trị của các biến. Chúng ta truyền vào biểu thức và dictionary values chứa giá trị của các biến để kiểm tra xem biểu thức có đúng hay không.
2. **find\_model**: Hàm này xác định tất cả các biến có trong biểu thức, sau đó tạo tất cả các kết hợp giá trị (True/False) của các biến. Với mỗi kết hợp, hàm sẽ đánh giá biểu thức và trả về mẫu giá trị nếu biểu thức đúng.
3. **Ví dụ sử dụng**: Biểu thức logic trong ví dụ là (A or B) and (not A or C). Chương trình sẽ kiểm tra tất cả các kết hợp giá trị của A, B, và C để tìm một mẫu giá trị khiến biểu thức đúng.

**Đầu vào:**

Kịch bản 1:

* Biểu thức: (A ∨ B) ∧ (¬A ∨ C)
* Các biến: A, B, C

Kịch bản 2:

* Biểu thức: (A ∧ B) ∨ (C ∧ ¬A)
* Các biến: A, B, C

Kịch bản 3:

* Biểu thức: A ∨ (B ∧ C)
* Các biến: A, B, C

Kịch bản 4:

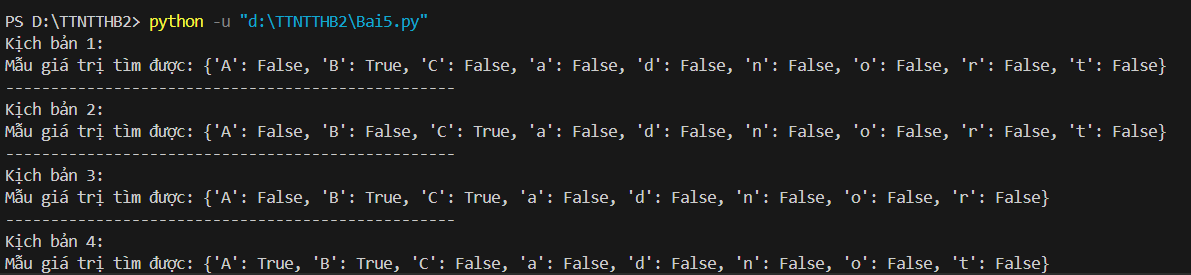
* Biểu thức: (A ∧ B) ∧ (¬C)
* Các biến: A, B, C

Kịch bản 5:

* Biểu thức: (A ∨ B) ∧ (¬A ∨ ¬C)
* Các biến: A, B, C

**Đầu ra:**

* Nếu có mẫu giá trị khiến biểu thức đúng, trả về mẫu giá trị dưới dạng dictionary, ví dụ: {A: False, B: True, C: True}.
* Nếu không có mẫu giá trị nào khiến biểu thức đúng, thông báo: "Không có mẫu giá trị nào làm biểu thức đúng."



A screen shot of a computer

Description automatically generated