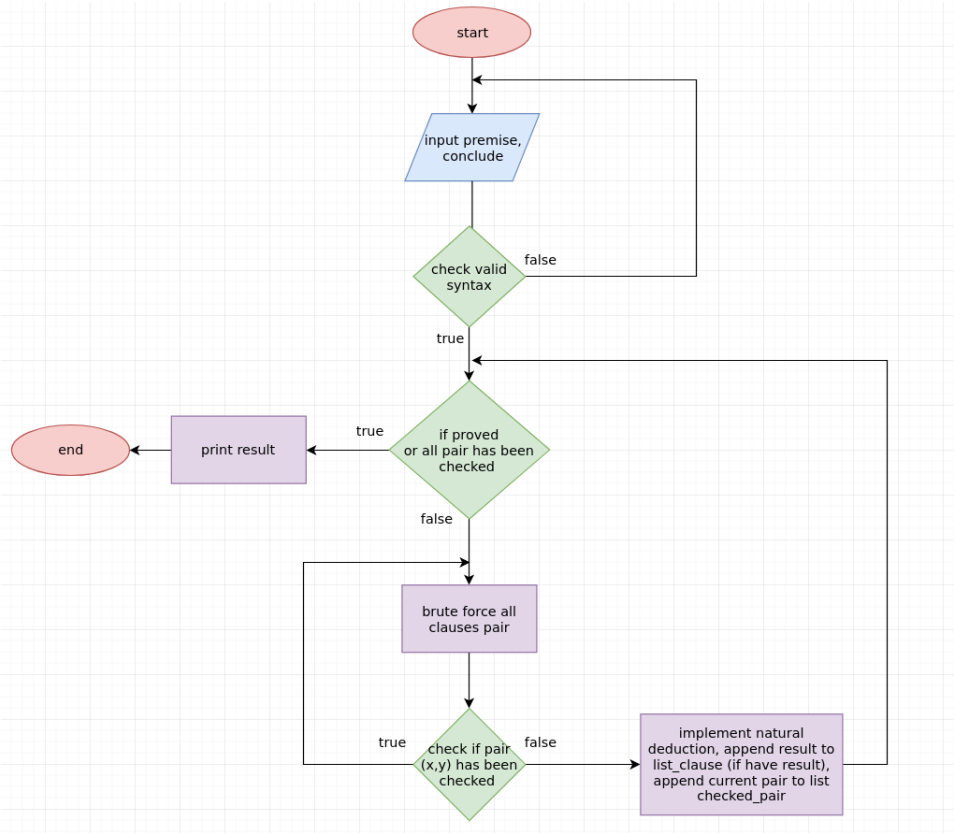
Đồ Án Logic Mệnh Đề

Họ tên	Mã số sinh viên	Đóng góp	Hoàn Thành
Đoàn Khuê	1612311	đồ án 1 (output), đồ án 2 (logic input output)	80%
Nguyễn Văn Linh	1612340	đồ án 1 (logic), đồ án 2 (input)	100%

Mục lục

Đồ án 1

Giải Thuật



Ý nghĩa các hàm

```
def check_clause_B_is_sub_clause_A_AND(clause_A, clause_B)
```

- Kiểm tra xem mệnh đề B (có dạng X.Y) có nằm trong mệnh đề A hay không
- Return boolen

```
def check_clause_B_is_sub_clause_A_OR(clause_A, clause_B):
```

- Kiểm tra xem mệnh đề B (có dạng X+Y) có nằm trong mệnh đề A hay không

- Return boolen

```
def check_clause_B_is_EQ_clause_A(clause_A, clause_B):
```

- Kiểm tra mệnh đề B có suy ra được mệnh đề A không
- Return boolen

```
def nCr(n, r):
```

- Tính số lượng các cặp mệnh đề khác nhau trong danh sách
- Return số lượng các cặp mệnh đề

```
def negative_clause(clause):
```

- Phủ định mệnh đề
- Return mệnh đề đã được phủ định

```
def check_valid_syntax(clause):
```

- Kiểm tra cú pháp mệnh đề có hợp lệ hay không
- Return boolen

```
def remove_double_negative(clause):
```

- Áp dụng luật EQ rút gọn mệnh đề có dạng phủ định của phủ định
- Return mệnh đề đã được rút gọn

```
def check_is_equal(clause_A, clause_B):
```

- Kiểm tra mệnh đề A có tương đương với mệnh đề B không
- Return boolen

```
def check_is_negative(clause_A, clause_B):
```

- Kiểm tra mệnh đề A có phải là dạng phủ định của mệnh đề B không
- Return boolen

```
def append_deduction_list(clause, proof):
```

- Thêm mệnh đề vào danh sách, cùng với dẫn chứng đã sử dụng để suy ra mệnh đề đó
- Return

```
def input_amount_premise():
```

- Nhập số lượng tiền đề
- Return

```
def input_list_premise(amount_premise):
```

- Nhập các tiền đề
- Return

```
def input_conclude():
```

- Nhập kết luận
- Return

```
def implement_CON_rule(clause_A, clause_B, idx_A, idx_B):
```

- Áp dụng Conjunction rule cho mệnh đề A và B
- Return

```
def check_HS_rule(clause_A, clause_B, idx_A, idx_B):
```

- Kiểm tra Hypothetical Syllogism rule giữa 2 mệnh đề A B, nếu tồn tại quan hệ thì áp dụng Hypothetical Syllogism rule
- Return

```
def natural_deduction(clause_A, clause_B, idx_A, idx_B):
```

- Hàm thực thi các luật Modus Ponens, Modus Tollens, Simplification, Addition
- Return

```
def check_all_pair():
```

- Hàm thực hiện vòng lặp duyệt toàn bộ mệnh đề có trong danh sách
- Return

```
def main():
```

- Hàm main của chương trình
- Return

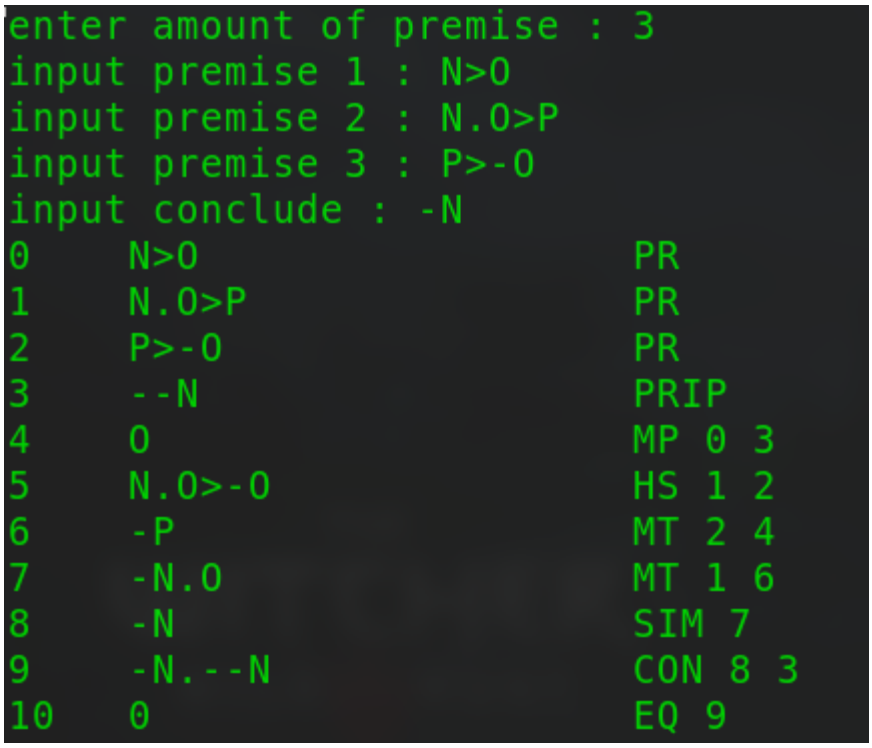
Cách sử dụng

- Cài đặt python
- Chạy chương trình bằng câu lệnh `python natural_deduction_logic.py`
- Nhập số lượng tiền đề
- Nhập tiền đề
- Nhập kết luận
- Chỉ sử dụng các chữ cái in hoa và các kí tự `(`, `)`, `-`, `.`, `+`, `>`, `=`

Kết quả đạt được

```
enter amount of premise : 4
input premise 1 : A+-B+-D
input premise 2 : E.F>D
input premise 3 : -A
input premise 4 : F.E
input conclude : -B
0    A+-B+-D                PR
1    E.F>D                  PR
2    -A                     PR
3    F.E                    PR
4    --B                    PRIP
5    -B+-D                  DS 0 2
6    A+-D                   DS 0 4
7    E.F                    EQ 3
8    -D                     DS 6 2
9    D                      MP 1 7
10   D.-D                   CON 9 8
11   0                      EQ 10
```

```
enter amount of premise : 1
input premise 1 : A+B>C.D
input conclude : A>D
0    A+B>C.D                PR
1    A                      PRCP
2    -D                     PRIP
3    A+B                    ADD 1
4    C.D                    MP 0 3
5    D                      SIM 4
6    D.-D                   CON 5 2
7    0                      EQ 6
```



Đồ án 2

Giải Thuật

- Biến đổi tất cả các câu thành dạng CNF
- Lấy phủ định kết luận, đưa vào KB
- Lặp
  - Nếu trong KB có chứa hai mệnh đề mâu thuẫn (ví dụ: `P` và `¬P`) thì trả về `True`
  - Sử dụng một biến mệnh đề để hợp giải:
    - Lấy tất cảcác câu chứa biến mệnh đề được chọn
    - Áp dụng luật hợp giải lên mọi cặp câu chứa khẳng định và phủ định của biến mệnh đề
  - Viết các câu kết quả mới và xoá các câu đã sử dụng
  - Lặp cho đến khi không còn biến mệnh đề nào có thể hợp giải được
- Trả về `False`

Ý nghĩa các hàm

```
to_cnf(list)
    return list
```

- Chuyển một list các câu về dạng CNF theo các bước sau (mỗi bước nằm trong một class:
  - Thêm dấu ngoặc theo đúng độ ưu tiên
  - Khai triển `=` thành `>` và `.`
  - Khai triển `>` thành `.` và `-`
  - Thực hiện De Morgan xử lí `-`
  - Phân phối

```
negate(string)
    return string
```

- Phủ định một câu, dùng để phủ định kết luận

```
split_and(list)
    return list
```

- Các câu có `^` thì tách làm hai câu

```
resolution(list)
    return boolean
```

- Thực hiện hợp giải

```
print_underline(list, string)
```

- Xuất ra KB với các câu chứa biến mệnh đề được gạch dưới

```
negate(string)
```

```
    return string
```

- Lấy phủ định của câu

```
add_brackets(string)
    return string
```

- Thêm ngoặc để sắp xếp theo thứ tự ưu tiên

```
get_result(object)
    return string
```

- Trả về kết quả của bước thực hiện

```
merge_items(object)
    return string
```

- Gộp các kết quả thành string hoàn chỉnh

```
replace_iff(string)
    return string
```

- Khai triển `=`
- Tương tự cho imp (`>`)

```
doing_demorgan(string)
    return string
```

- Đưa `-` vào trong

```
distribute(object)
    return boolean
```

- Phân phối giữa `.` và `+`

```
reducing_and(string)
    return string
```

- Đơn giản hóa `.`
- Tương tự cho `+`

```
merging(object)
```

- Gộp kết quả

Cách sử dụng

Tạo một file `input.txt` với nội dung là các mệnh đề và dòng cuối cùng là kết luận

Ví dụ:

```
A > B + D
E . A > -B
F . E
```

Giữa các kí tự ngoại trừ dấu '-' phải có dấu cách

Sau đó chạy chương trình bằng lệnh `python project2.py` ta có kết quả:

```
Input clauses
['A > B + D', 'E . A > -B', 'F . E']
Knowledge base formated as CNF
['- A + B + D', '- E + -B + - A', 'E . F']
Splitted and
['- A + B + D', '- E + -B + - A', 'E', 'F']
```

Vì chức năng vẫn chưa hoàn thành hết nên kết quả không in ra được hết các câu.

Kết quả đạt được

Chức năng	Hoàn thành
Đọc được input từ file	100%
Biến đổi các câu thành dạng CNF	100%
Tách `.` thành các câu khác nhau	100%
Lấy phủ định kết luận	50%
Hợp giải	30%

Tham Khảo

- https://en.wikipedia.org/wiki/Natural_deduction
- https://www.iep.utm.edu/nat-ded/
- https://proofs.openlogicproject.org/
- https://github.com/ldkrsi/cnf_py
- https://github.com/khamkarajinkya/Davis-Putnam-Logemann-Loveland-Algorithm