

**TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM  
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG  
KHOA CÔNG NGHỆ THÔNG TIN**



**BÀI TẬP LỚN MÔN CẤU TRÚC RỜI RẠC**

# **LẬP BẢNG CHÂN TRỊ BẢNG GIẢI THUẬT REVERSE POLISH NOTATION**

*Người hướng dẫn:* **THẦY NGUYỄN HUỖNH MINH DUY**

*Người thực hiện:* **NGUYỄN TÔN ĐIỀN - 52000643**

**Lớp : 20050201**

**Khoá : 24**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2021**

**TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM  
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG  
KHOA CÔNG NGHỆ THÔNG TIN**



**BÀI TẬP LỚN MÔN CẤU TRÚC RỜI RẠC**

# **LẬP BẢNG CHÂN TRỊ BẢNG GIẢI THUẬT REVERSE POLISH NOTATION**

Người hướng dẫn: **THẦY NGUYỄN HUỲNH MINH DUY**

Người thực hiện: **NGUYỄN TÔN ĐIỀN - 52000643**

Lớp : **20050201**

Khoá : **24**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2021**

## TÓM TẮT

Đây là bài báo cáo bài tập lớn môn Cấu trúc rời rạc, qua bài tập này, em đã được ôn lại thuật toán Reverse Polish Notation đã học từ môn Cấu trúc dữ liệu và giải thuật và làm thêm thao tác mới là tạo bảng chân trị từ postfix. Trong quá trình làm bài, sử dụng ngôn ngữ mới là python, em được học thêm về itertools và hiểu thêm bảng chân trị sinh ra từ itertools là mảng các tuples, xác định được điều này mới có thể ghép bảng chân trị qua hàm mergeTable() tự định nghĩa.

## MỤC LỤC

MỤC LỤC.....	1
DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ .....	2
CHƯƠNG 1 – GIỚI THIỆU .....	3
CHƯƠNG 2 – CƠ SỞ LÝ THUYẾT .....	4
2.1 Giải thuật Reverse Polish Notation (RPN) (phần này trích từ 1) .....	4
2.1.1 Lý thuyết.....	4
2.1.2 Ví dụ.....	4
2.2 Các phép tính logic cơ bản của bảng chân trị .....	6
CHƯƠNG 3 – GIẢI THÍCH CODE.....	7
3.1 Case 1: $R (P \& Q)$ .....	8
3.2 Case 2: $\sim P (Q \& R) > R$ .....	10
CHƯƠNG 4 – KẾT QUẢ 5 TEST CASE .....	13
4.1 Case 1: $R (P \& Q)$ .....	13
4.2 Case 2: $\sim P (Q \& R) > R$ .....	14
4.3 Case 3: $P (R \& Q)$ .....	15
4.4 Case 4: $(P > Q) \& (Q > R)$ .....	16
4.5 Case 5: $(P \sim Q) > \sim P = (P (\sim Q)) > \sim P$ .....	17
CHƯƠNG 5 – TÀI LIỆU THAM KHẢO .....	18

## **DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ**

### **DANH MỤC HÌNH**

Hình 2.1. Ví dụ chuyển đổi infix biểu thức số sang postfix	5
Hình 2.2. Ví dụ chuyển đổi infix biểu thức logic sang postfix	5
Hình 3.1. Infix to postfix case 2	10
Hình 4.1 Kết quả case 1	13
Hình 4.2 Kết quả case 2	14
Hình 4.3 Kết quả case 3	15
Hình 4.4 Kết quả case 4	16
Hình 4.5 Kết quả case 5	17

### **DANH MỤC BẢNG**

Bảng 1.1. Bảng tiến trình làm việc	3
Bảng 2.1 Các phép tính logic cơ bản	6
Bảng 4.1 Kết quả case 1	13
Bảng 4.2 Kết quả case 2	14
Bảng 4.3 Kết quả case 3	15
Bảng 4.4 Kết quả case 4	16
Bảng 4.5 Kết quả case 5	17

## CHƯƠNG 1 – GIỚI THIỆU

Bài làm trên được làm cá nhân, được làm trong 3 ngày:

1/12/2021	Hoàn thành code
2/12/2021 – 3/12/2021	Hoàn thành báo cáo

Bảng 1.1. Bảng tiến trình làm việc

Giới thiệu các chương sau:

- Chương 2, giới thiệu lí thuyết giải thuật Reverse Polish và lí thuyết các phép tính logic cơ bản của bảng chân trị
- Chương 3, giải thích code qua test case 1 và 2
- Chương 4, dán kết quả của 5 testcase
- Chương 5, trích dẫn

## CHƯƠNG 2 – CƠ SỞ LÝ THUYẾT

### 2.1 Giải thuật Reverse Polish Notation (RPN) (phần này trích từ 1)

#### 2.1.1 Lý thuyết

Là giải thuật mà phép toán theo sau toán hạng. Ví dụ, thay vì  $3 + 4$  như cách viết bình thường, qua giả thuật RPN ta sẽ viết thành  $3\ 4\ +$ . Cách viết này giúp ta tính toán nhanh hơn vì ít thông tin phải lưu trữ hơn

Áp dụng để chuyển đổi từ infix sang postfix:

- Bước 1, duyệt chuỗi infix từ trái qua phải
- Bước 2, nếu đó là toán hạng, thêm nó vào chuỗi postfix
- Bước 3, nếu duyệt đến kí tự “(”, push kí tự đó vào stack
- Bước 4, nếu duyệt đến kí tự “)”, nghĩa là ta có một phép kết hợp cần được tính:
  - +) pop ra từ stack và thêm phép toán vừa được pop ấy vào chuỗi postfix, lặp lại cho đến khi tìm thấy kí tự “(“
  - +) xóa kí tự “(“
- Bước 5, nếu một phép toán được tìm thấy:
  - +) liên tục pop từ trong stack những phép toán mà có độ ưu tiên lớn hơn hoặc bằng phép toán vừa tìm thấy, và thêm những phép toán vừa pop ấy vào chuỗi postfix
  - +) thêm phép toán được tìm thấy vào stack
- Bước 6, nếu không còn phần tử nào trong chuỗi infix, pop stack cho đến khi stack hết phần tử, và thêm những phần tử mới bị pop vào chuỗi postfix

#### 2.1.2 Ví dụ

- Xét infix là biểu thức số học bình thường:  $a - (b + c * d) / e$

<u>ch</u>	<u>Stack (bottom to top)</u>	<u>postfixExp</u>
a		a
-	-	a
(	-(	a
b	-(	a b
+	-( +	a b
c	-( +	a b c
*	-( + *	a b c
d	-( + *	a b c d
)	-( +	a b c d *
	-(	a b c d * +
	-	a b c d * +
/	- /	a b c d * +
e	- /	a b c d * + e
		a b c d * + e / -

Hình 2.1. Ví dụ chuyển đổi infix biểu thức số sang postfix

Kết quả: a b c d \* + e / -

- Xét infix là biểu thức toán logic: R|(P&Q)

<u>ch</u>	<u>Stack (bottom to top)</u>	<u>postfixExp</u>
R		R
		R
(	(	R
P	(	RP
&	(&	RP
Q	(&	RPQ
)	(	RPQ&
		RPQ&
		RPQ&

Hình 2.2. Ví dụ chuyển đổi infix biểu thức logic sang postfix

Kết quả: RPQ&|



## 2.2 Các phép tính logic cơ bản của bảng chân trị

T: True, F: False

P	Q	Not P	P and Q	P or Q	P implies Q	P bi implies Q
T	T	F	T	T	T	T
T	F	F	F	T	F	F
F	T	T	F	T	T	F
F	F	T	F	F	T	T

Bảng 2.1 Các phép tính logic cơ bản

## CHƯƠNG 3 – GIẢI THÍCH CODE

Hàm `Infix2Postfix(Infix)` có logic đã được giải thích (chương 2) nên đây là phần giải thích chung khi hiện thực code:

```
def Infix2Postfix(Infix):
    Postfix = ''
    stack = []
    ###BUOC 1, DUYET CHUOI TU TRAI SANG PHAI
    for ch in Infix:
        ###BUOC 2, THEM TOAN HANG
        if(ch >= 'A' and ch <= 'Z'):
            Postfix = Postfix + ch
        ###BUOC 3 , THEM KI TU '(' VAO STACK
        if(ch == '('):
            stack.append(ch)
        ###BUOC 4, KHI GAP KI TU ')'
        if(ch == ')'):
            ##POP STACK LUU VAO POSTFIX CHO DEN KHI GAP '('
            while(stack[-1] != '('):
                Postfix = Postfix + stack.pop()
            ##XOA KI' TU '('
            stack.pop()
        ###BUOC 5, KHI GAP PHEP TOAN
        if(ch == '~' or ch == '&' or ch == '|' or ch == '>' or ch == '='):
            ## POP NHUNG PHEP TOAN CO DO UU TIEN CAO HON THEM VAO POSTFIX,
            DUNG LAI KHI STACK RONG HOAC GAP '('
            while(len(stack) != 0 and
                  stack[-1] != '(' and
                  precedence(ch) <= precedence(stack[-1])):
                Postfix = Postfix + stack.pop()
            ## THEM PHEP TOAN VAO STACK
            stack.append(ch)

        ###BUOC 6, POP STACK LUU VAO POSTFIX CHO DEN KHI STACK RONG
        while(len(stack) != 0):
            Postfix = Postfix+stack.pop()

    return Postfix
```

### 3.1 Case 1: R|(P&Q)

- Sau hàm Infix2Postfix(Infix), ta có được kết quả RPQ&| (chương 2).
- Đưa kết quả trên vào hàm Postfix2Truthtable(Postfix):

Tạo list tên charList[] để lưu các toán hạng, sau khi lưu xong sẽ có các toán hạng trùng lặp trong biểu thức gốc và chưa được xếp theo bảng chữ cái, nên không thể tạo Truthtable được. Chính vì thế gọi hàm set(charList) để xóa phần tử trùng lặp rồi sort list charList theo thứ tự A, B, C, ...

```
#####
charList = [] #luu cac operand -> khoi tao truth table

for ch in Postfix:
    if(ch >= 'A' and ch <= 'Z'):
        charList.append(ch)

charList = list(set(charList)) #set de xoa phan tu trung lap -> chuyen lai
thanh list de sort
charList.sort() #sort de xep theo thu tu A, B, C, ...

Truthtable = generate_table(len(charList))
```

```
def generate_table(n):
    return list(itertools.product([False, True], repeat= n))
```

Sau bước này ta đã khởi tạo được Truthtable của 3 toán hạng P, Q, E

Dùng stack[] để thực hiện các phép toán trên postfix

```
for ch in Postfix:
    if(ch >= 'A' and ch <= 'Z'):
        stack.append(ch)

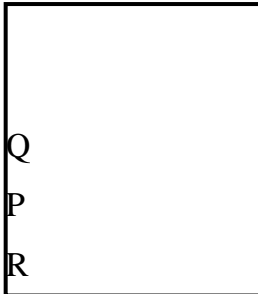
    if ch == '&':
        arg1 = stack.pop()
        arg2 = stack.pop()
        charList.append(''.join(arg2+'&'+arg1))
        stack.append(''.join(arg2+'&'+arg1))
        tmpList = [a[charList.index(arg1)] and a[charList.index(arg2)] for
a in Truthtable]
```

```

        Truthtable = mergeTable(tmpList,Truthtable)
    if ch == '|':
        arg1 = stack.pop()
        arg2 = stack.pop()
        charList.append(''.join(arg2+'|'+arg1))
        stack.append(''.join(arg2+'|'+arg1))
        tmpList = [a[charList.index(arg1)] or a[charList.index(arg2)] for
a in Truthtable]
        Truthtable = mergeTable(tmpList,Truthtable)

```

Từ RPQ&|:



Stack khởi tạo

Sau khi gặp ‘&’ pop từ stack ra P và Q rồi thực hiện phép toán, đưa ra kết quả rồi merge cái cột các kết quả đó vào Truthtable thông qua hàm mergeTable()

```

def mergeTable(r,Truthtable):
    i = 0
    res = []
    for tup in Truthtable:
        li = list(tup)
        li.append(r[i])
        i+=1
        res.append(tuple(li))

    return res

```



Stack

Tiếp tục gặp '|' pop ra từ stack (P&Q) rồi thực hiện phép toán, đưa ra kết quả rồi merge vào Truthtable như trên



Stack

Cuối cùng return Truthtable cần tìm

### 3.2 Case 2: $\sim P|(Q\&R)>R$

- Sau hàm Infix2Postfix(Infix), ta có được kết quả  $P\sim QR\&|R>$

<u>ch</u>	<u>Stack (bottom to top)</u>	<u>postfixExp</u>
~	~	
P	~	P
		P~
(	(	P~
Q	(	P~Q
&	(&	P~Q
R	(&	P~QR
)		P~QR&
>	>	P~QR&
R		P~QR& >

Hình 3.1. infix to postfix case 2

- Sau đó, ta đưa postfix  $P\sim QR\&|R>$  vào hàm Postfix2Truthtable(Postfix):

Vì lần này có phép toán kéo theo và phủ định nên ta code thêm

```
def lImplies(p,q):
    if p==True:
        return q
    else:
        return True
```

```
stack = [] #dung de tinh toan
```

```

for ch in Postfix:
    if(ch >= 'A' and ch <= 'Z'):
        stack.append(ch)

    if ch == '~':
        arg = stack.pop()
        charList.append('').join('~'+arg))
        stack.append('').join('~'+arg))
        tmpList = [ not a[charList.index(arg)] for a in Truthtable]
        Truthtable = mergeTable(tmpList,Truthtable)
    if ch == '&':
        arg1 = stack.pop()
        arg2 = stack.pop()
        charList.append('').join(arg2+'&'+arg1))
        stack.append('').join(arg2+'&'+arg1))
        tmpList = [a[charList.index(arg1)] and a[charList.index(arg2)] for
a in Truthtable]
        Truthtable = mergeTable(tmpList,Truthtable)
    if ch == '|':
        arg1 = stack.pop()
        arg2 = stack.pop()
        charList.append('').join(arg2+'|'+arg1))
        stack.append('').join(arg2+'|'+arg1))
        tmpList = [a[charList.index(arg1)] or a[charList.index(arg2)] for
a in Truthtable]
        Truthtable = mergeTable(tmpList,Truthtable)
    if ch == '>':
        arg1 = stack.pop()
        arg2 = stack.pop()
        charList.append('').join(arg2+'>'+arg1))
        stack.append('').join(arg2+'>'+arg1))
        tmpList = [lImplies(a[charList.index(arg2)],
a[charList.index(arg1)]) for a in Truthtable]
        Truthtable = mergeTable(tmpList,Truthtable)
    if ch == '=':
        arg1 = stack.pop()
        arg2 = stack.pop()
        charList.append('').join(arg2+'>'+arg1))
        stack.append('').join(arg2+'>'+arg1))
        tmpList = [a[charList.index(arg1)] == a[charList.index(arg2)] for
a in Truthtable]
        Truthtable = mergeTable(tmpList,Truthtable)

```

Từ  $P \sim QR \& |R >$  :

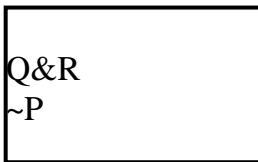


Stack khởi tạo

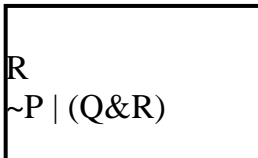
Sau khi gặp '~':



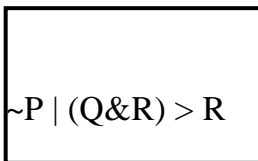
Sau khi gặp '&':



Sau khi gặp '|':



Sau khi gặp '>':



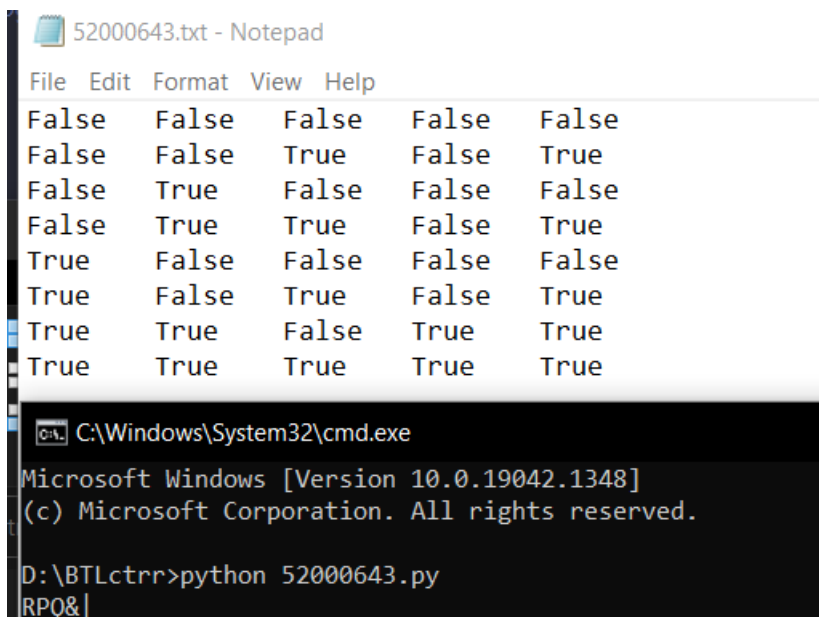
Trong quá pop stack ra để tính toán như trên, Truthtable luôn được tính và cập nhật thêm cột kết quả, và sau khi kết thúc quá trình ta có thể return được Truthtable mong muốn.

## CHƯƠNG 4 – KẾT QUẢ 5 TEST CASE

### 4.1 Case 1: $R|(P \& Q)$

Input	Output				
$R (P \& Q)$	P	Q	R	$P \& Q$	$R (P \& Q)$
	False	False	False	False	False
	False	False	True	False	True
	False	True	False	False	False
	False	True	True	False	True
	True	False	False	False	False
	True	False	True	False	True
	True	True	False	True	True
	True	True	True	True	True

Bảng 4.1 Kết quả case 1



The image shows a Notepad window titled '52000643.txt - Notepad' with the following text:

```
File Edit Format View Help
False False False False False
False False True False True
False True False False False
False True True False True
True False False False False
True False True False True
True True False True True
True True True True True
```

Below the Notepad window is a Windows Command Prompt window titled 'C:\Windows\System32\cmd.exe' showing the following text:

```
Microsoft Windows [Version 10.0.19042.1348]
(c) Microsoft Corporation. All rights reserved.

D:\BTLctr>python 52000643.py
RPQ&|
```

Hình 4.1 Kết quả case 1



## 4.2 Case 2: $\sim P|(Q \& R) > R$

Input	Output						
$\sim P (Q \& R) > R$	P	Q	R	$\sim P$	$(Q \& R)$	$\sim P (Q \& R)$	$\sim P (Q \& R) > R$
	False	False	False	True	False	True	False
	False	False	True	True	False	True	True
	False	True	False	True	False	True	False
	False	True	True	True	True	True	True
	True	False	False	False	False	False	True
	True	False	True	False	False	False	True
	True	True	False	False	False	False	True
	True	True	True	False	True	True	True

Bảng 4.2 Kết quả case 2

```

52000643.txt - Notepad
File Edit Format View Help
False False False True False True False
False False True True False True True
False True False True False True False
False True True True True True True
True False False False False False True
True False True False False False True
True True False False False False True
True True True False True True True

C:\Windows\System32\cmd.exe
D:\BTLctrr>python 52000643.py
P~QR&|R>

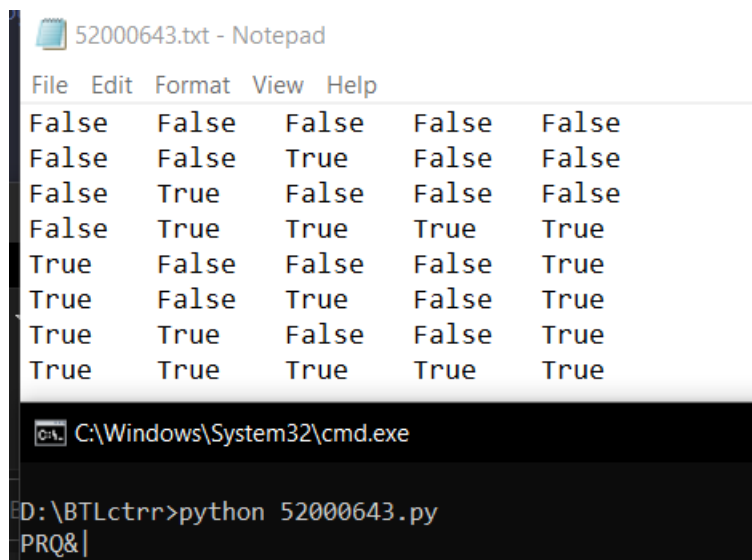
```

Hình 4.2 Kết quả case 2

### 4.3 Case 3: $P|(R \& Q)$

Input	Output				
$P (R \& Q)$	P	Q	R	$R \& Q$	$P (R \& Q)$
	False	False	False	False	False
	False	False	True	False	False
	False	True	False	False	False
	False	True	True	True	True
	True	False	False	False	True
	True	False	True	False	True
	True	True	False	False	True
	True	True	True	True	True

Bảng 4.3 Kết quả case 3



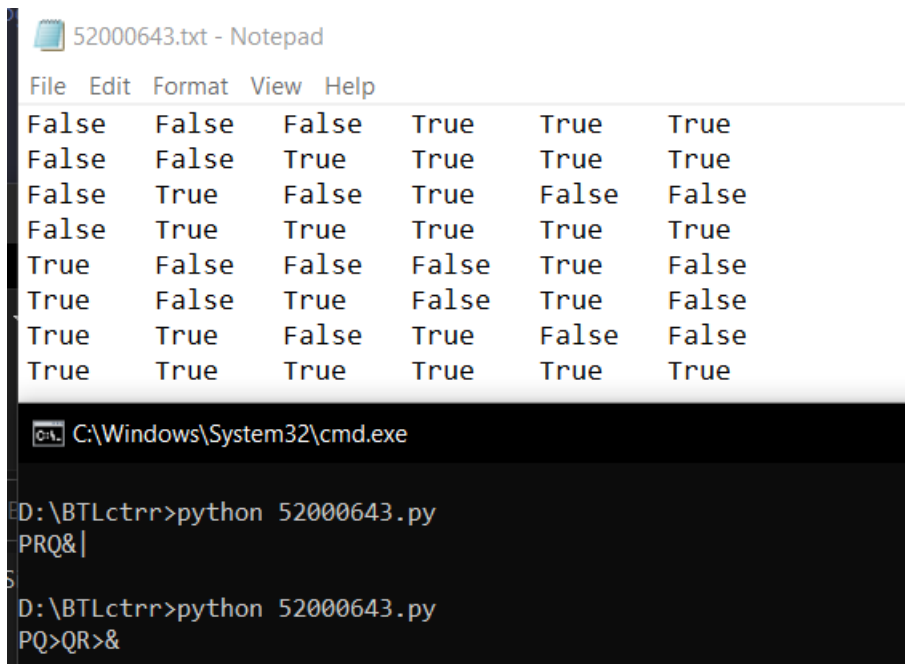
The image shows a Notepad window titled "52000643.txt - Notepad" with a menu bar (File, Edit, Format, View, Help) and a text area containing the same truth table data as in the table above. Below the Notepad window is a Windows command prompt window titled "C:\Windows\System32\cmd.exe" showing the command `python 52000643.py` being executed in a directory `D:\BTLctrr`. The prompt shows the output `PRQ&`.

Hình 4.3 Kết quả case 3

#### 4.4 Case 4: $(P \supset Q) \& (Q \supset R)$

Input	Output					
$(P \supset Q) \& (Q \supset R)$	P	Q	R	$P \supset Q$	$Q \supset R$	$(P \supset Q) \& (Q \supset R)$
	False	False	False	True	True	True
	False	False	True	True	True	True
	False	True	False	True	False	False
	False	True	True	True	True	True
	True	False	False	False	True	False
	True	False	True	False	True	False
	True	True	False	True	False	False
	True	True	True	True	True	True

Bảng 4.4 Kết quả case 4



The image shows a Notepad window titled "52000643.txt - Notepad" with the following text:

```
File Edit Format View Help
False False False True True True
False False True True True True
False True False True False False
False True True True True True
True False False False True False
True False True False True False
True True False True False False
True True True True True True
```

Below the Notepad window is a command prompt window titled "C:\Windows\System32\cmd.exe" showing the execution of a Python script:

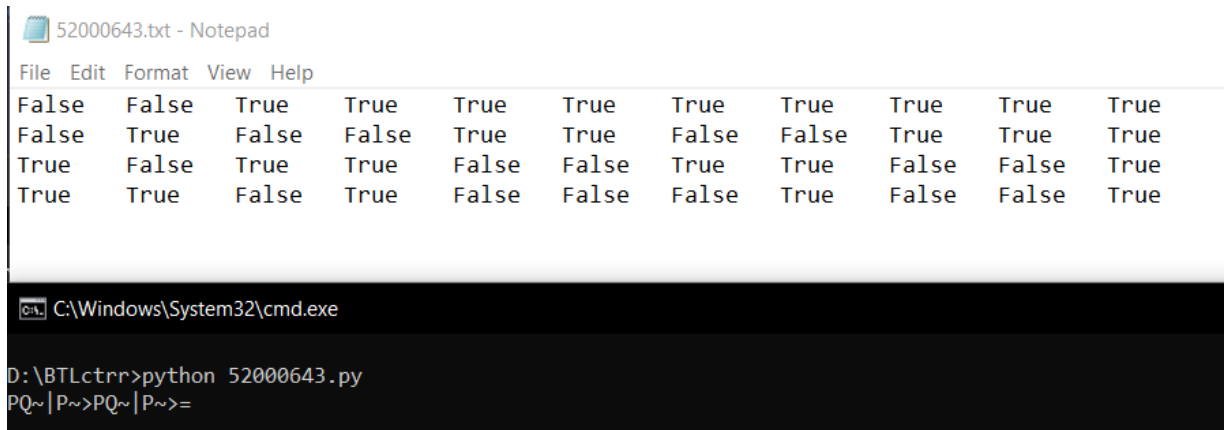
```
D:\BTLctrr>python 52000643.py
PRQ&|
D:\BTLctrr>python 52000643.py
PQ>QR>&
```

Hình 4.4 Kết quả case 4

#### 4.5 Case 5: $(P|\sim Q)>\sim P = (P|(\sim Q))>\sim P$

Input	Output				
$(P \sim Q)>\sim P$ = $(P (\sim Q))>\sim P$	P	Q	A = $(P \sim Q)>\sim P$	B = $(P (\sim Q))>\sim P$	A $\Leftrightarrow$ B
	False	False	True	True	True
	False	True	True	True	True
	True	False	False	False	False
	True	True	False	False	False

Bảng 4.5 Kết quả case 5



Hình 4.5 Kết quả case 5

## **CHƯƠNG 5 – TÀI LIỆU THAM KHẢO**

[1] Data Structures and Algorithms, Stack and Queues (2015), by Mr. Aaron Tan Tuck Choy, and Dr. Low Kok Lim, National University of Singapore