

## 一、 操作手冊

### 【程式功能說明】

繪製四變數布林代數式之卡諾圖，並找出 Minimum SOP 所有可能解。

### 【程式使用流程】

1. 將要讀取的檔案名稱設為 `input.txt`，並與其他檔案放在同一資料夾中。

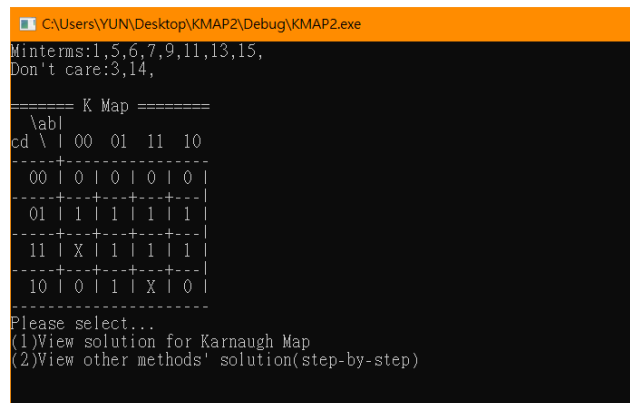
`input.txt` 格式：

將所要進行化簡的四變數布林代數式以下方範例之 SOP 格式表示。

\*Don't care 部分請寫在 ( ) 內。

範例： $ad + a'bc + c'd + (a'b'cd) + (abcd')$

2. 開啟 `simulator.exe`，執行畫面如下：



```
C:\Users\YUN\Desktop\KMAP2\Debug\KMAP2.exe
Minterms:1,5,6,7,9,11,13,15,
Don't care:3,14,

===== K Map =====
\abl
cd \ | 00 01 11 10
-----+-----+-----+-----+
00 | 0 | 0 | 0 | 0 | 0 |
-----+-----+-----+-----+
01 | 1 | 1 | 1 | 1 | 1 |
-----+-----+-----+-----+
11 | X | 1 | 1 | 1 | 1 |
-----+-----+-----+-----+
10 | 0 | 0 | 1 | X | 0 |
-----+-----+-----+-----+
Please select...
(1)View solution for Karnaugh Map
(2)View other methods' solution(step-by-step)
```

3. 請依指示輸入：
  - 1 → 觀看卡諾圖之解法與式子化簡結果
  - 2 → 觀看其他解法 (Quine McClasky 或 Petrick's Method) 與式子化簡結果 (會顯示詳細步驟)
4. 若選擇 1，執行畫面如下：



```
< SOLUTION >
===== K Map =====
\abl
cd \ | 00 01 11 10
-----+-----+-----+-----+
00 | 0 | 0 | 0 | 0 | 0 |
-----+-----+-----+-----+
01 | 1 | 1 | 1 | 1 | 1 |
-----+-----+-----+-----+
11 | X | 1 | 1 | 1 | 1 |
-----+-----+-----+-----+
10 | 0 | 0 | 1 | X | 0 |
-----+-----+-----+-----+

group 1:6,7,14,15
Simplification of group 1->bc
group 2:1,3,5,7,9,11,13,15
Simplification of group 2->d
F(a,b,c,d) = bc+d
請按任意鍵繼續 . . .
```

5. 若輸入 2，執行畫面如下：（請依照指示操作以繼續下一步）

```
C:\Users\YUN\Desktop\KMAP2\Debug\KMAP2.exe
* Step3:The remaining unchecked terms are prime implicants:
[P0] 6,7,14,15->bc
[P1] 1,3,5,7,9,11,13,15->d
Draw the Prime Implicant Chart.
< Prime Implicant Chart >
=====
               1   5   6   7   9  11  13  15
-----
[P0]           6,7,14,15,      x  x              x
[P1] 1,3,5,7,9,11,13,15,  x  x      x  x  x  x  x
=====

Please select a method...
(1)Quine MacClasky Method
(2)Petrick's Method
```

```
C:\Users\YUN\Desktop\KMAP2\Debug\KMAP2.exe
* Step1: Group the minterms according to the number of 1's.

<Group 0>
-----
<Group 1>
1      0001      x
-----
<Group 2>
3      0011      x
5      0101      x
6      0110      x
9      1001      x
-----
<Group 3>
7      0111      x
11     1011      x
13     1101      x
14     1110      x
-----
<Group 4>
15     1111      x
-----
Press ENTER to see the next step...
```

6. 執行至 Step3 時，請依指示輸入，選擇欲使用之解法：

1 → 使用 Quine McClasky

2 → 使用 Petrick's Method

7. 若輸入 1：Quine McClasky，執行畫面如下：（請依照指示操作以繼續下一步）

```
C:\Users\YUN\Desktop\KMAP2\Debug\KMAP2.exe
* Step 2:
Comparison of adjacent groups:
Combine terms that have the same '-' and differ one in the number of 1's.
Check off the combined terms.

<Group 0>
-----
<Group 1>
1      0001      y
-----
<Group 2>
3      0011      y
5      0101      y
6      0110      x
9      1001      y
-----
<Group 3>
7      0111      x
11     1011      x
13     1101      x
14     1110      x
-----
<Group 4>
15     1111      x
-----
<Group 5>
1,3    00-1      x
1,5    0-01      x
1,9    -001      x
-----
Press ENTER to see the next step...
```

```
C:\Users\YUN\Desktop\KMAP2\Debug\KMAP2.exe

Apply Quine McClasky Method.
< Prime Implicant Chart >
=====
          1   5   6   7   9  11  13  15
-----
[P0]      6,7,14,15,   x   x   x   x   x
[P1] 1,3,5,7,9,11,13,15, x   x   x   x   x
=====

Press ENTER to see the next step...

Minterm1 is only covered by P1.
P1 has to be chosen. Delete P1 and the other minterms it covers.

< Prime Implicant Chart >
=====
          6
-----
[P0]      6,7,14,15, x
=====

Press any key to see the next step...

Minterm6 is only covered by P0.
P0 has to be chosen. Delete P0 and the other minterms it covers.

< Prime Implicant Chart >
=====
-----
=====

Press any key to see the next step...

F(a,b,c,d) = bc+d
請按任意鍵繼續 . . .
```

\*若有多組解，將接續使用Petrick's Method找出所有可能解，如下：

```
C:\Users\YUN\Desktop\KMAP2\Debug\KMAP2.exe

< Prime Implicant Chart >
=====
          11
-----
[P2]  9,11,13,15,   x
[P3] 10,11,14,15,   x
=====

Press any key to see the next step...

The remaining chart is CYCLIC, apply Petrick's Method to find solution.

< Equation >
( P2 + P3 )

Press ENTER to see the next step...

< SOLUTION >
(Pick the terms with fewest prime implicants.)
(Combine with Quine McClasky's solution.)
F(a,b,c,d) = b'cd'+c'd+ad+ab

or

F(a,b,c,d) = b'cd'+c'd+ac+ab
請按任意鍵繼續 . . .
```

8. 若輸入 2，執行畫面如下：(請依照指示操作以繼續下一步)

```
C:\Users\YUN\Desktop\KMAP2\Debug\KMAP2.exe

Apply Petrick's Method.
< Prime Implicant Chart >

=====
              1   5   6   7   9  11  13  15
-----
[P0]          6,7,14,15,      x   x           x
[P1] 1,3,5,7,9,11,13,15,  x   x       x   x   x   x   x
=====

Press ENTER to see the next step...

< Equation >
  ( P0 ) ( P0 + P1 ) ( P1 )

Press ENTER to see the next step...

= ( P0 P1 )

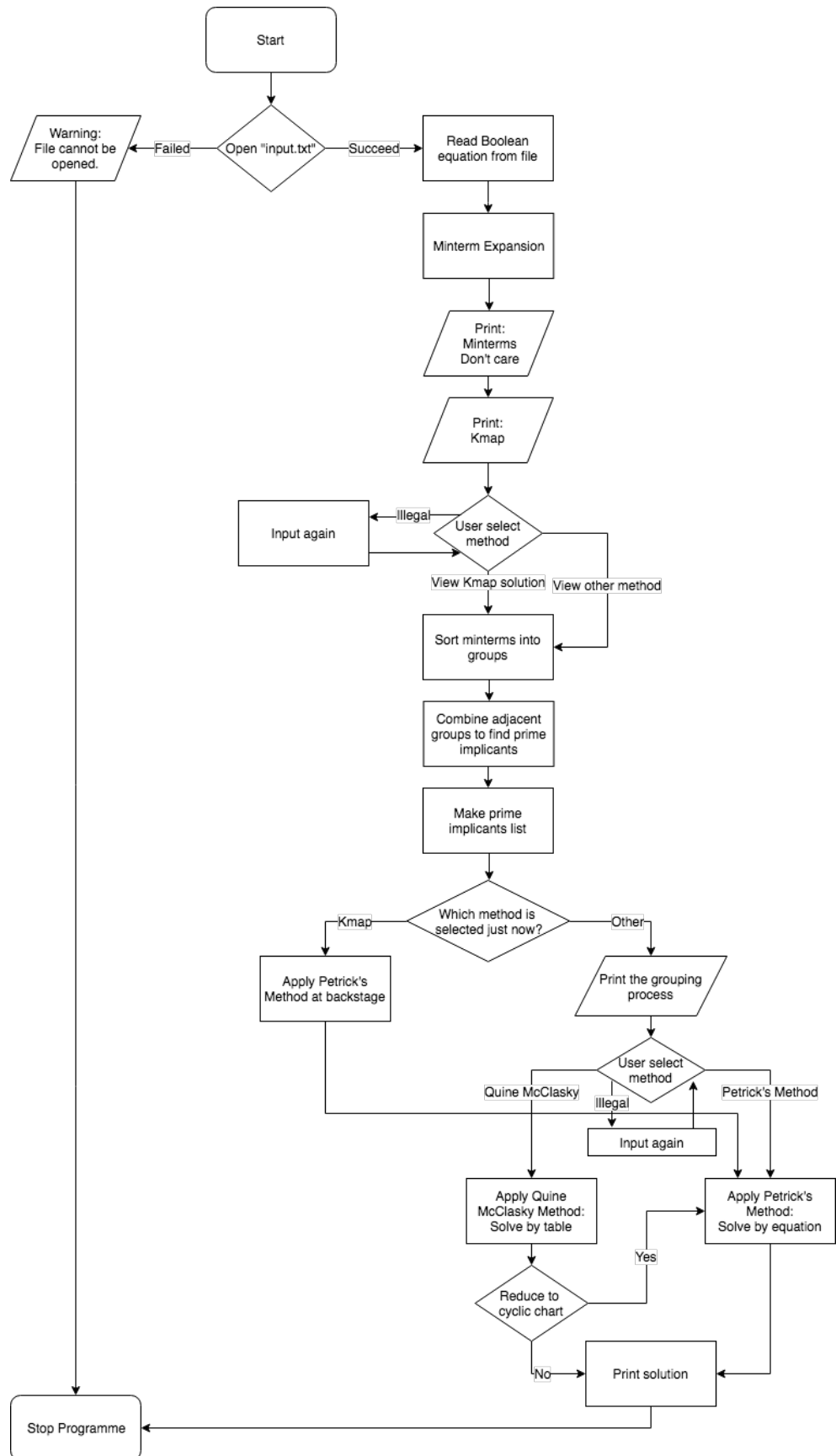
Press ENTER to see the next step...

< SOLUTION >
(Pick the terms with fewest prime implicants.)
F(a,b,c,d) = bc+d

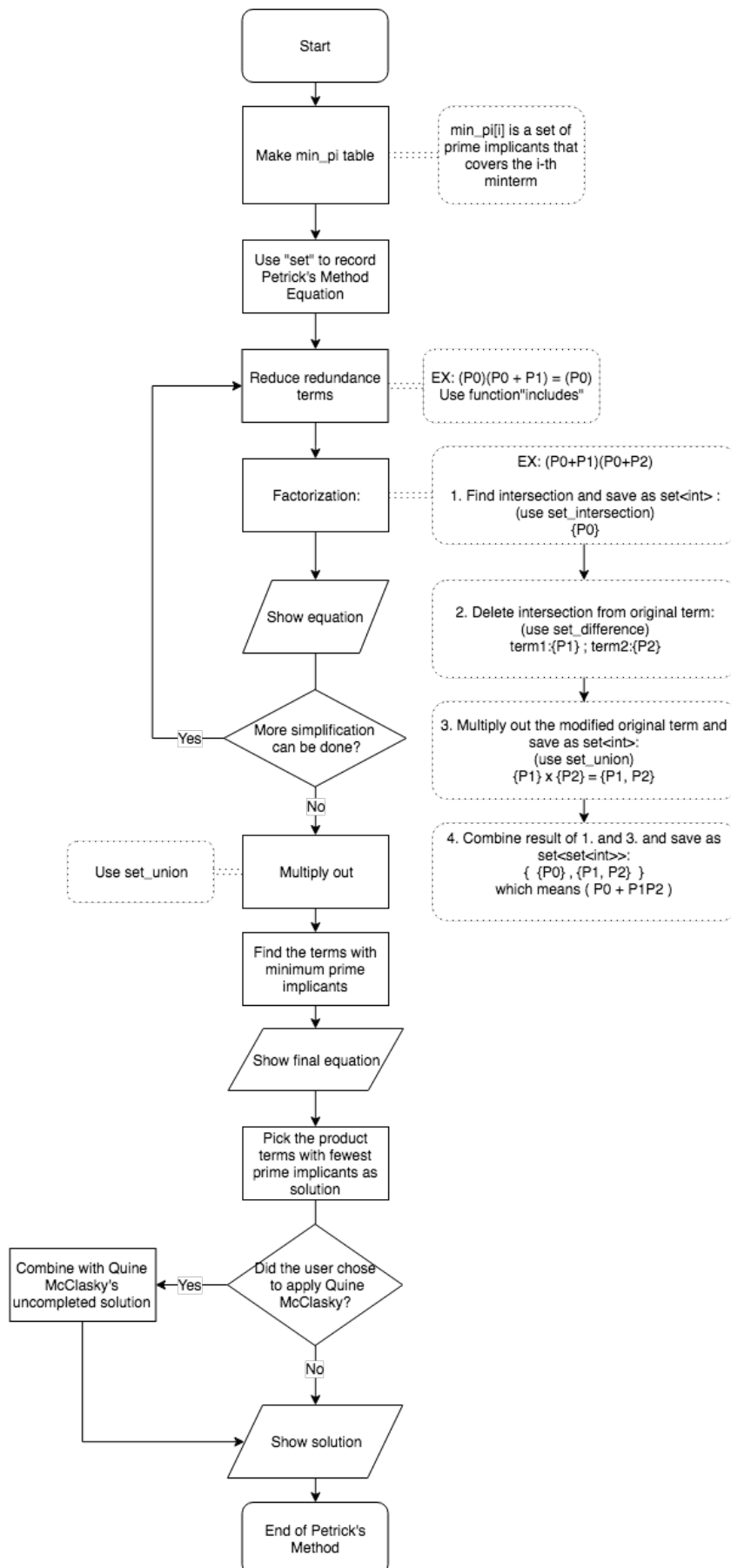
請按任意鍵繼續 . . .
```

## 二、 程式撰寫流程（已省略繁瑣微小的細節）

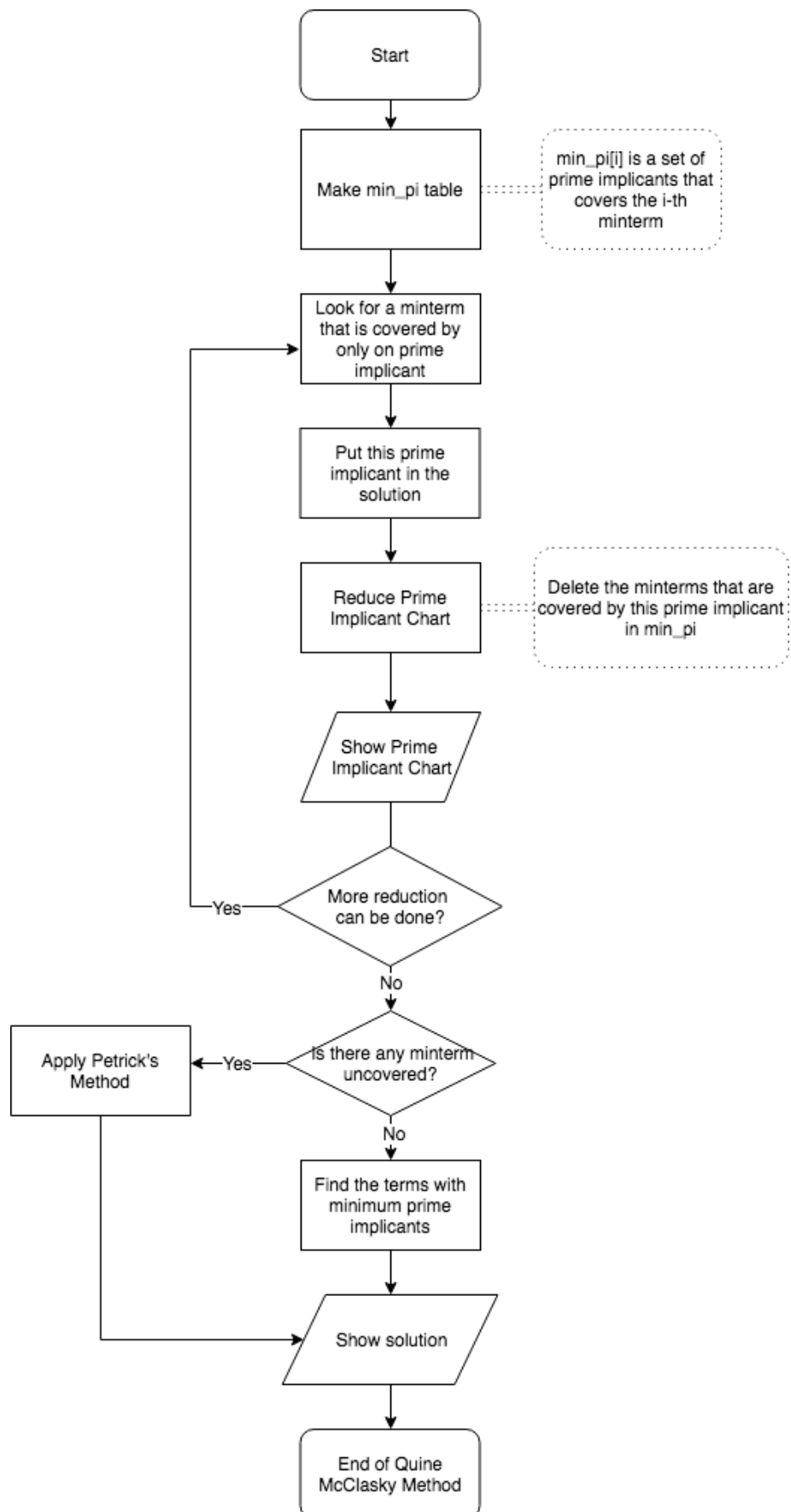
### 1. 主程式 Source.cpp



## 2. Petricks Method



### 3. Quine McClasky Method



### 三、 作業心得

這次的作業，我選擇不使用直接在卡諾圖上畫圈，而採用了 Quine McClasky 和 Petrick's Method，一來是因為我認為卡諾圖本身是設計給人類使用的方便工具，而 Quine McClasky 和 Petrick's Method 是較有系統的，只是前置作業是必須一一將 Prime Implicants 做組合，複雜度對人類來說比較大，但電腦運算夠快，較適合這種系統性的方法。二來，在變數數量改變時，礙於表格設計方式迥異，我們所寫的四變數卡諾圖的程式也就失去了效用，但另外兩種方法卻不受影響，因此我決定試著寫出這兩種方法的程式，即便這次的作業沒有要求，未來我也能使用這支自己寫的程式去解多樣化的題型。

其中 Petrick's Method 是最花時間的。它並不如手寫時那麼簡單，實際開始構思後我才想到，如： $(P_1+P_2)(P_1+P_3) = P_1 + P_2P_3$  不是只要設 int 變數來存 1、2、3 代表  $P_1$ 、 $P_2$ 、 $P_3$ ，然後找交集差集就能輕鬆了事，因為繼續化簡下去，X、Y、Z 都可能變成  $P_n$  的組合：

如  $(X+Y)(X+Z) = (X+YZ)$  變成

$$\text{狀況① } (P_1 + \boxed{P_2+P_3}) (\boxed{P_2+P_3} + \boxed{P_4+P_5}) = (P_2+P_3 + P_1 (P_4+P_5))$$

$$\text{狀況② } (P_1 + \boxed{P_2+P_3}) (P_1 + \boxed{P_4+P_5}) = (P_1 + (P_2+P_3) (P_4+P_5))$$

遇到這樣複雜的狀況時，便會出現許多問題：

狀況①  $X = P_2+P_3$ ，無法單純用 int 代表 X

狀況② 產生新的 POS，但這個方法就是要把 POS 化成 minimum SOP，這樣會沒完沒了。

由於種種問題，這個部分我想了非常久，最後想到用 set 來存，而且 set 的元素的唯一性也相當方便，但麻煩的是設計出來的是整個式子會變成三層的 set，我在寫程式時被搞混了很多次，有大半的時間都在處理這個部分。

自己手動解卡諾圖時，可以很直接地在圖上圈出正解，但將過程用程式寫出來卻相對麻煩很多。但設計程式時，由於無法像直觀地想怎麼在紙上做記號就直接動筆，而必須一步一步地用程式碼呈現，整個解題過程中必須將邏輯徹底理清，因此設計完之後，縱然我相信效能優化還有很大的進步空間，但透過不斷地除錯修正，也能夠讓人對解題方法更加融會貫通。



#### 四、 Source Code 說明

主程式	內容	
Source.cpp (參考 二、1. Source.cpp 流程圖)	使用 C++ library	
	Library	備註
	#include <iostream>	
	#include <fstream>	讀檔使用到 ifstream
	#include <cmath>	Minterm Expansion 中計算 $2^b \rightarrow \text{pow}(2,b)$
	#include <iomanip>	使用格式化輸出 $\rightarrow \text{setw}(x)$
	#include <algorithm>	取 set 交集 $\rightarrow \text{set\_intersection}$ 取 set 差集 $\rightarrow \text{set\_difference}$ 取 set 聯集 $\rightarrow \text{set\_union}$
	#include <iterator>	使用上方三個函式時需用到 inserter
	#include <string>	
	#include <vector>	
	#include <set>	
	詳細說明	
	自建函式	
	<pre>void mintermExpansion (int min_sop[16], int output, int minterm, int term[4], int b) {...}</pre>	
	<pre>void getSolution (vector&lt;Cmbterms&gt; &amp;pi_list, vector&lt;int&gt; minterms, const int min_sop[16], const char &amp;choice, const char &amp;choice2) {...}</pre>	

自建 Class	內容	
Cmbterms.h Cmbterms.cpp (class Cmbterms 的	使用 C++ library	
	Library	備註
	#include <iostream>	

header file 和 implementation file)	#include <iomanip>	使用格式化輸出 → setw(x)
	#include <set>	
	#include <vector>	
	#include <algorithm>	取set交集 → set_intersection 取set差集 → set_difference 取set聯集 → set_union
	#include <iterator>	使用上方三個函式時需用到 inserter
	詳細說明	
	為 implicants 的合併與分組而設計	
	主要 Data members 和 Member functions	
	set<int> minterm;	儲存包含的 minterms
	set<int> pos_1;	儲存二進位表示法中 1 在第幾個 bit
	set<int> pos_x;	儲存二進位表示法中 – (don't care) 在第幾個 bit
	bool checked;	記錄是否已被合併
	Bool can Combine (const Cmbterms&c2) {...}	回傳是否能和 c2 合併
	Cmbterms operator+(Cmbterms &c2) {...}	回傳和 c2 合併後的結果

副程式	內容	
print.h print.cpp (格式化輸出函式 的 header file 和 implementation file)	使用 C++ library	
	Library	備註
	#include <iostream>	
	#include <sstream>	printPiChart() 中將 Cmbterms 包含的 minterms 轉為 string 時使用
	#include <iomanip>	
	#include <string>	

	#include <vector>	
	#include <set>	
	詳細說明	
	將格式化輸出函式獨立以使主程式功能單純化	
	void printSet(set<int> set)	印出 Petrick's Method 式子中的 $P_n$
	void printSset(set< set<int> > sset)	印出 Petrick's Method 式子中的 $P_n$ 組成的 product term
	void printSum(set< set< set<int> > > sum)	印出 Petrick's Method 完整的式子
	void printKmap(const int min_sop[16])	
	void printGroup(const vector< vector<Cmbterms> > &G)	印出將 implicants 合併、分組的過程和結果
	void printPiChart(const vector< Cmbterms > &pi_list, const vector<int> &minterms, const vector<int> &isPruned_pi)	印出 Quine McClasky 和 Petrick's Method 的 Prime Implicant Chart