網安實務 Hw1

網工所碩一 309552005 吳偉誠

Part A

Q1.

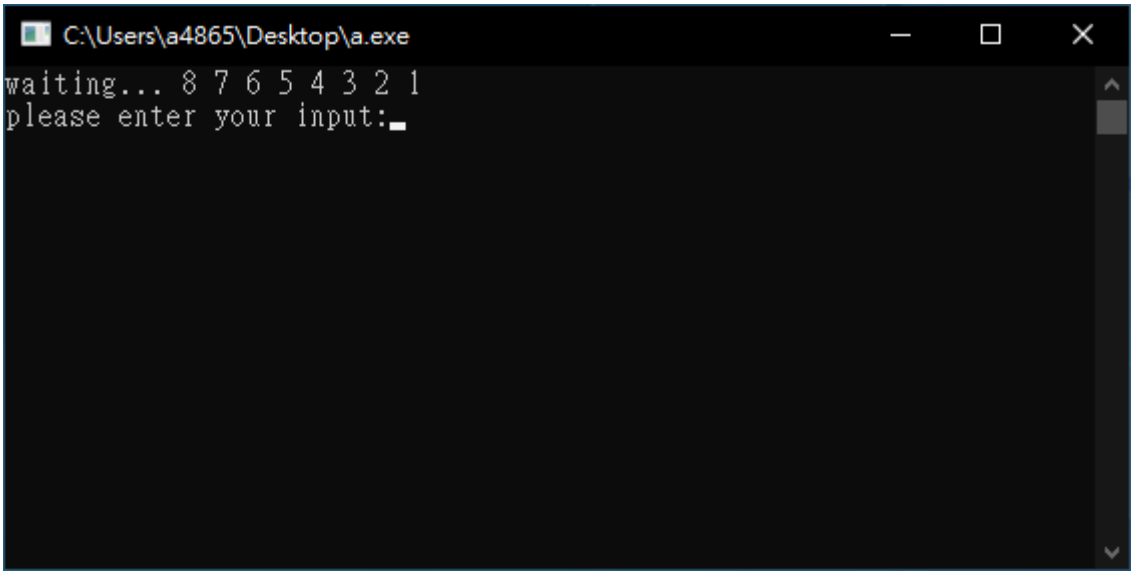
前言

當我用筆電開a.exe時，它有跳出我缺少3樣的.dll檔，但當我載完後，卻跳出以下畫面：

> 本來是有跳出我缺了三個dll檔，安裝後再開就跳出這個

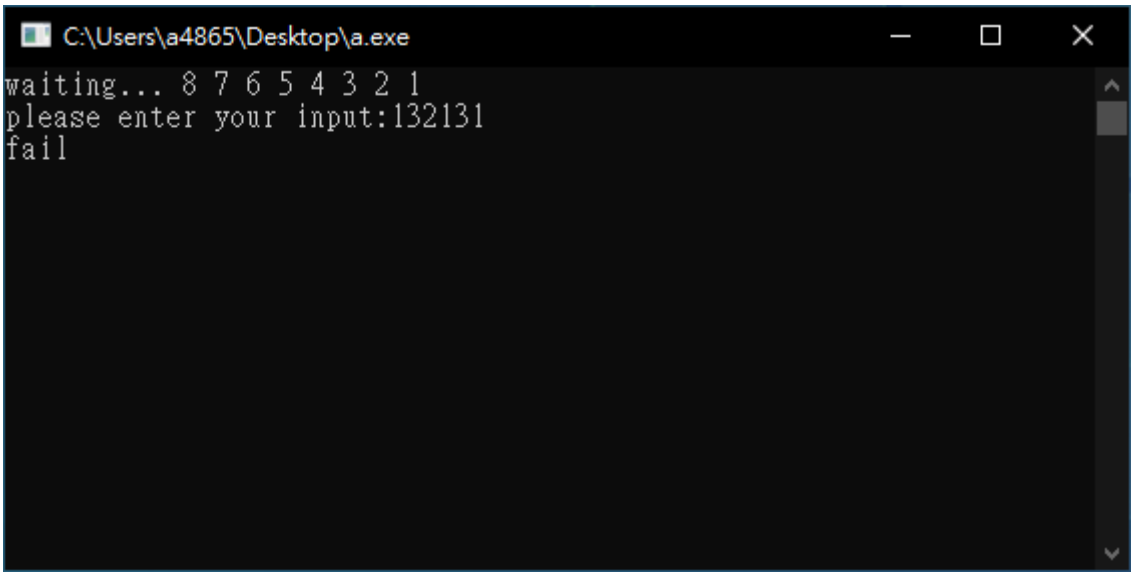
於是我就換實驗室的電腦跑看看，結果就可以了！
(不過助教後來有給修正版的，但都已截圖完，只剩做報告就沒去重做新的)

先正常的開啟程式看它如何跑：



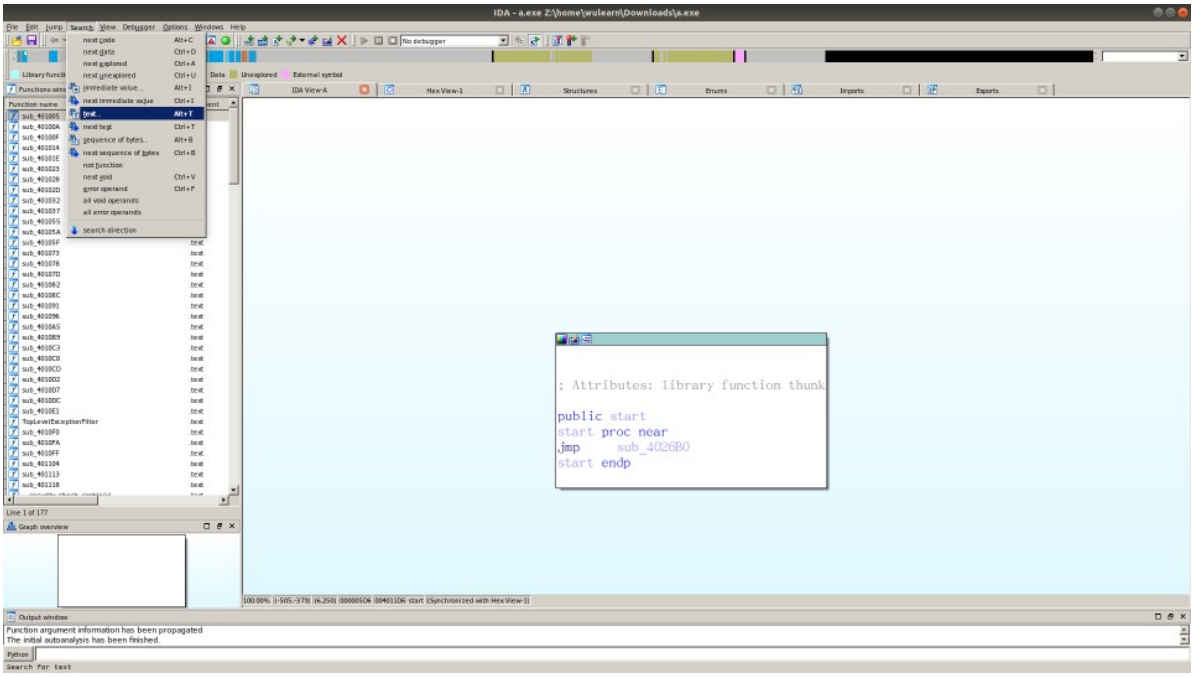
> 發現倒數完8秒後，會要求我們輸入答案

輸入錯誤後會出現以下畫面：

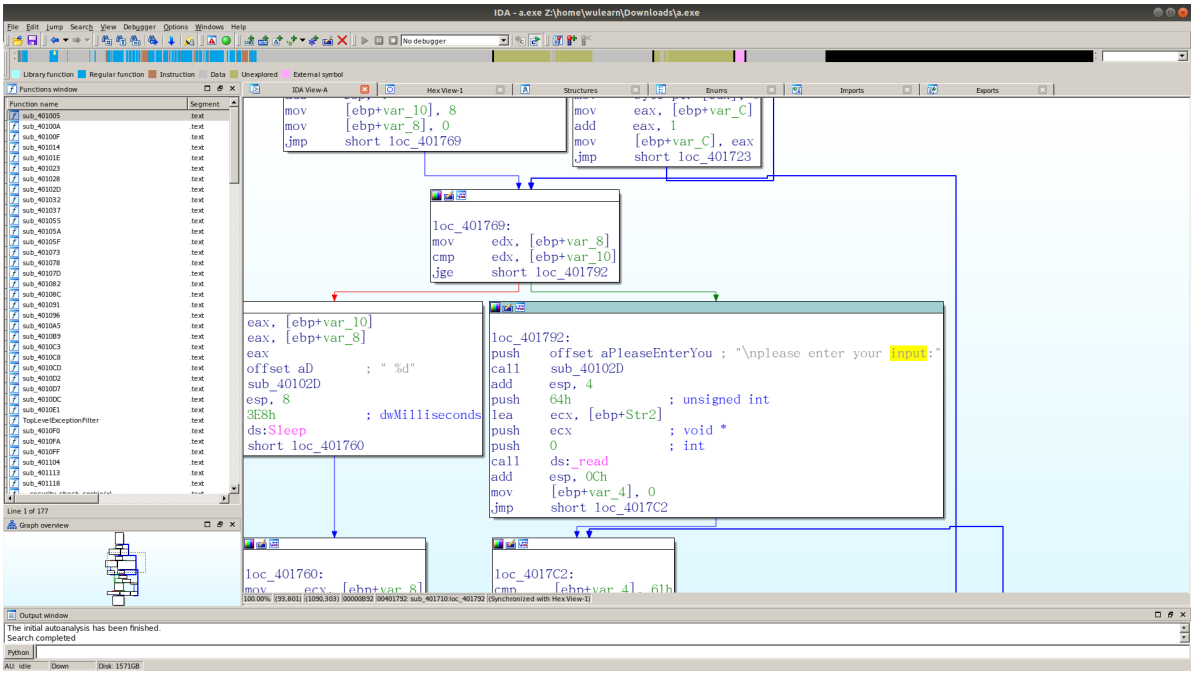


解題過程

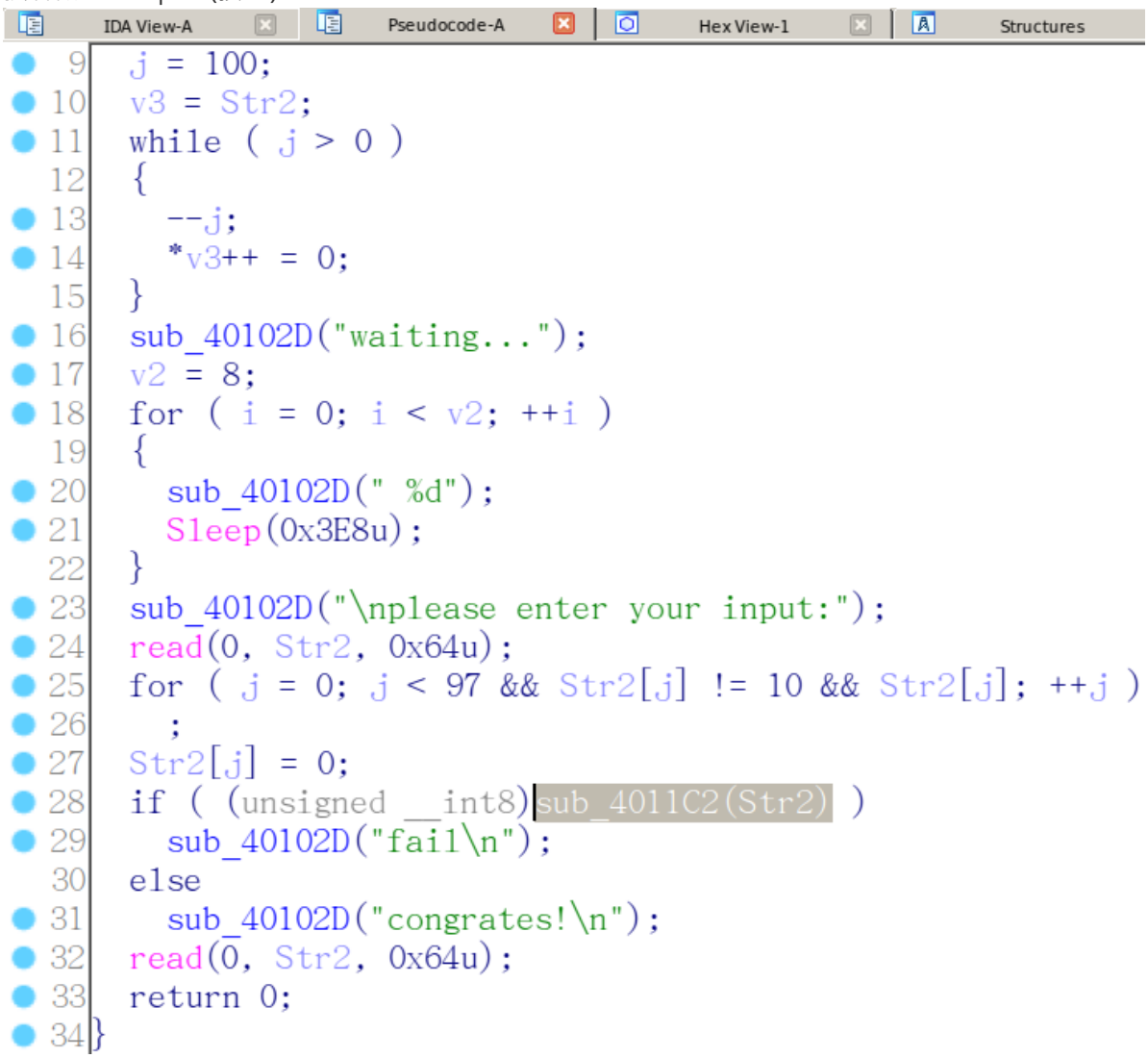
因為Ghidra反編譯後，我覺得太難看懂了(後面會貼那部份，其他流程跟以下類似就不截了)，所以用到上學期學到的工具IDA來trace：



> 流程基本上跟Ghidra一樣，先利用搜尋String的方式來找到該關鍵function的位置



接著做decompiler(按F5)：



> 大致分析可以知道：
>
> - Str2是我們輸入進去的值
> - 接著做一個for迴圈用於找new line的index
> - 並將該index值改成0(做字串結尾的處理（null）)

- 然後把輸入值丟到sub_4011C2函式裡
  - 如果回傳`False`:則顯示"fail"
  - 如果回傳`True`:則顯示"congrates!"

再深追sub_4011C2函式裡：

```
int __cdecl sub_4011C2(char *Str2)
{
  return sub_401690(Str2);
}
```

> 回傳sub_401690函式的值

sub_401690函式：
Ghidra下這邊的code是長這樣：

```
local_18 = 0x464d4149;
local_14 = 0x47414c;
local_10 = 0xffffffe0;
local_8 = 0;
local_c = 8;
while (local_8 < local_c) {
  if (*(char *)((int)&local_18 + local_8) != '\0') {
    *(char *)((int)&local_18 + local_8) = *(char *)((int)&local_18 + local_8) - (char)local_10;
  }
  local_8 = local_8 + 2;
}
iVar1 = strcmp((char *)&local_18,param_1);
return iVar1 != 0;
```

其存的值都是放在記憶體位置
但就沒去深追了，當下就直接換IDA來看這邊的差異
IDA:

```
1  bool __cdecl sub_401690(char *Str2)
2  {
3    char Str1[4]; // [esp+0h] [ebp-14h]
4    int v3; // [esp+8h] [ebp-Ch]
5    int v4; // [esp+Ch] [ebp-8h]
6    int v5; // [esp+10h] [ebp-4h]
7
8    strcpy(Str1, "IAMFLAG");
9    v3 = -32;
10   v5 = 0;
11   v4 = 8;
12   while ( v5 < v4 )
13   {
14     if ( Str1[v5] )
15       Str1[v5] -= v3;
16     v5 += 2;
17   }
18   return strcmp(Str1, Str2) != 0;
19 }
```

可以發現比Ghidra還好懂while裡面做的事：

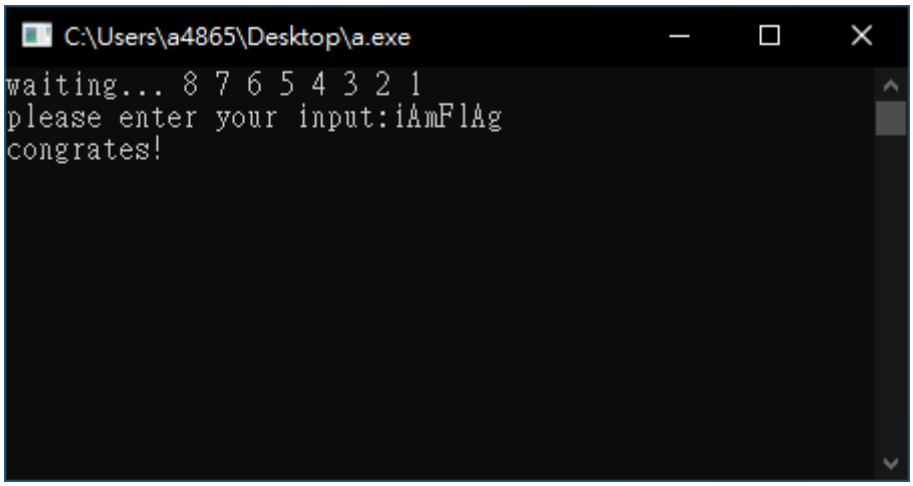
- 把字串"IAMFLAG"的奇數位的值+32 (這個意思就是大寫轉小寫)

Dec	Hex	Char
64	40	@
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z

Dec	Hex	Char
96	60	`
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z

> 也就是while完的值變成"iAmFlAg"
> 然後去驗證我們的輸入是否等於"iAmFlAg"

知道輸入後(`iAmFlAg`)就可以解開此題：



Part B

Q2.

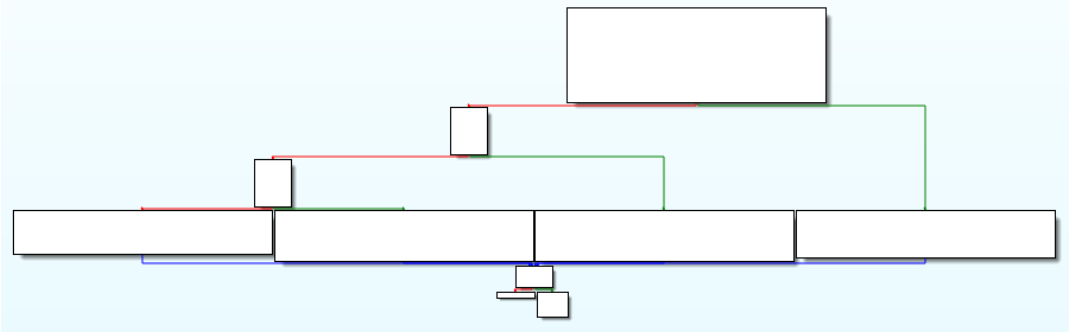
`Control Flow Flattening`：主要是將if-else語句替換成while語句，然後透過switch來控制Flow，這樣就能模糊basic block之間的前後關係。

Easy Program

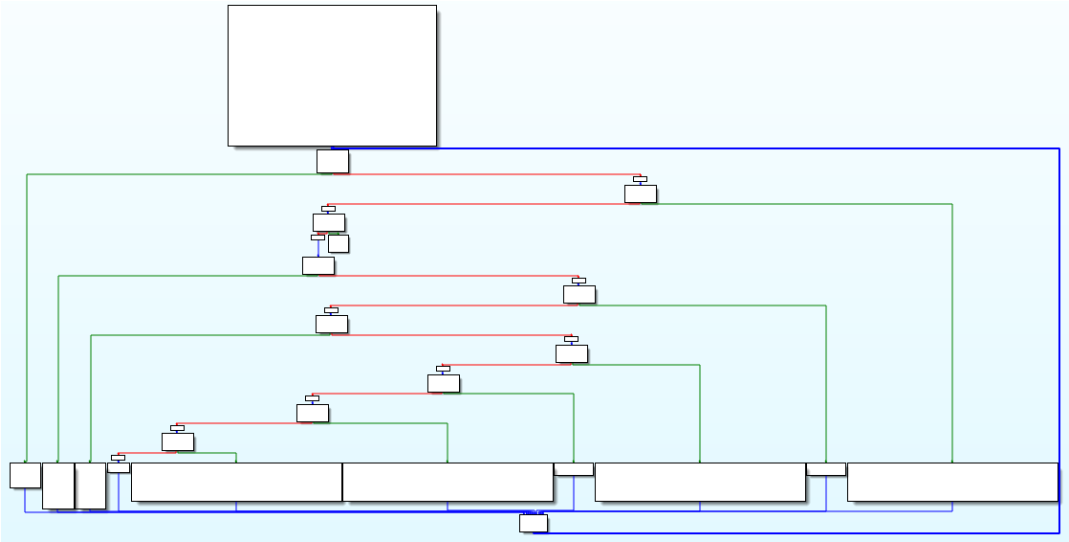
這邊我用C++寫個簡單的程式：判斷輸入的年份是否為閏年。
簡單來說：用到了三層if-else判斷式。

Assembly & 反編譯

- 首先來看Control Flow Graph上的差異：
  - 原：



- Flattening後：



> 可以看到整個程式結構從本來很直觀的if-else判斷式，被大幅的打亂掉了(可以從後面知道是變成了while＆switch的關係)。

- Discompiler後的Code:
  - 原(只截圖重點部份)：

```
std::operator<<<std::char_traits<char>>(&std::cout, "Enter a year: ", envp);
std::istream::operator>>(&std::cin, &v15);
if ( v15 & 3 )
{
  v11 = std::ostream::operator<<(&std::cout, v15);
  v13 = std::operator<<<std::char_traits<char>>(v11, " is not a leap year.", v12);
  std::ostream::operator<<(v13, &std::endl<char,std::char_traits<char>>);
}
else if ( (signed int)v15 % 100 )
{
  v8 = std::ostream::operator<<(&std::cout, v15);
  v10 = std::operator<<<std::char_traits<char>>(v8, " is a leap year.", v9);
  std::ostream::operator<<(v10, &std::endl<char,std::char_traits<char>>);
}
else
{
  if ( (signed int)v15 % 400 )
  {
    v6 = std::ostream::operator<<(&std::cout, v15);
    v5 = std::operator<<<std::char_traits<char>>(v6, " is not a leap year.", v7);
  }
  else
  {
    v3 = std::ostream::operator<<(&std::cout, v15);
    v5 = std::operator<<<std::char_traits<char>>(v3, " is a leap year.", v4);
  }
  std::ostream::operator<<(v5, &std::endl<char,std::char_traits<char>>);
}
return 0;
```

- Flattening後(只截圖重點部份)：

```
std::operator<<<std::char_traits<char>>(&std::cout, "Enter a year: ", envp);
std::istream::operator>>(&std::cin, &v20);
v22 = (signed int)v20 % 4;
v19 = -1589081681;
while ( 1 )
{
  while ( 1 )
  {
    while ( v19 == -1589081681 )
    {
      v3 = -380399592;
      if ( !v22 )
        v3 = -532830563;
      v19 = v3;
    }
    if ( v19 != -1571049527 )
      break;
    v6 = std::ostream::operator<<(&std::cout, v20);
    v8 = std::operator<<<std::char_traits<char>>(v6, " is a leap year.", v7);
    v19 = 356274438;
    std::ostream::operator<<(v8, &std::endl<char,std::char_traits<char>>);
  }
  if ( v19 == -1220852565 )
    break;
  switch ( v19 )
  {
    case -1196059895:
      v5 = 1999593273;
      if ( !((signed int)v20 % 400) )
        v5 = -1571049527;
      v19 = v5;
      break;
    case -778428681:
      v19 = -1220852565;
      break;
    case -532830563:
      v4 = 370713042;
      if ( !((signed int)v20 % 100) )
        v4 = -1196059895;
      v19 = v4;
      break;
    case -380399592:
      v15 = std::ostream::operator<<(&std::cout, v20);
      v17 = std::operator<<<std::char_traits<char>>(v15, " is not a leap year.", v16);
      v19 = -1220852565;
      std::ostream::operator<<(v17, &std::endl<char,std::char_traits<char>>);
      break;
    case 356274438:
      v19 = -778428681;
      break;
    case 370713042:
      v12 = std::ostream::operator<<(&std::cout, v20);
      v14 = std::operator<<<std::char_traits<char>>(v12, " is a leap year.", v13);
      v19 = -778428681;
      std::ostream::operator<<(v14, &std::endl<char,std::char_traits<char>>);
      break;
    case 1999593273:
```

> 反編譯後的code邏輯基本上跟Source Code大同小異。
> 但經過Flattening後的，可以發現被轉成有while跟switch的code，就變得更加難以知道其成邏輯關係。

Q3.

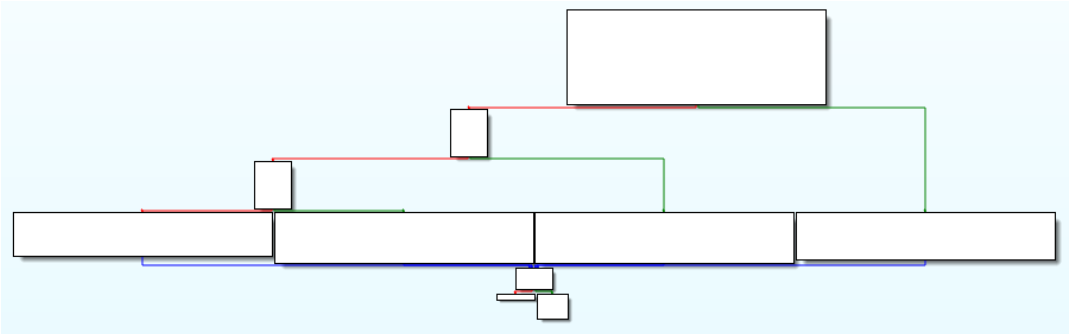
`Bogus Control Flow`：主要是在一個簡單的運算中外包好幾層if-else的判斷，讓逆向的難度上升。

Easy Program

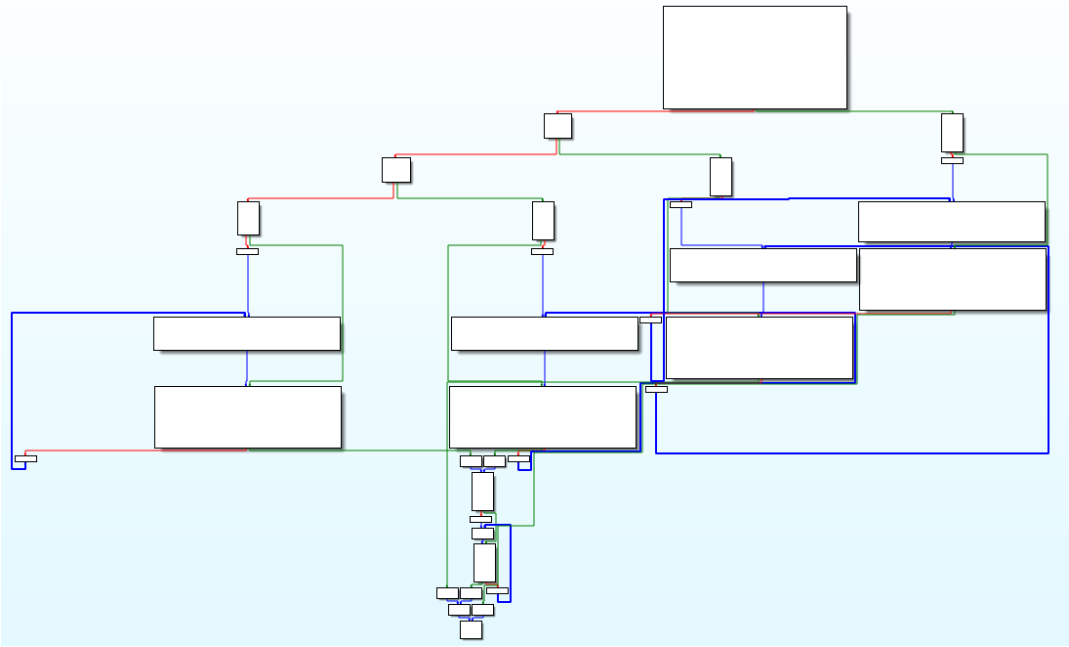
跟Q2使用同支程式。

Assembly & 反編譯

- 首先來看Control Flow Graph上的差異：
  - 原：



  - Bogus後：



> 可以看到整個程式結構從本來的三層if-else，多了許多意義不明的path與判斷式。

- Discompiler後的Code:

- 原(只截圖重點部份)：

```
std::operator<<<std::char_traits<char>>(&std::cout, "Enter a year: ", envp);
std::istream::operator>>(&std::cin, &v15);
if ( v15 & 3 )
{
  v11 = std::ostream::operator<<(&std::cout, v15);
  v13 = std::operator<<<std::char_traits<char>>(v11, " is not a leap year.", v12);
  std::ostream::operator<<(v13, &std::endl<char,std::char_traits<char>>);
}
else if ( (signed int)v15 % 100 )
{
  v8 = std::ostream::operator<<(&std::cout, v15);
  v10 = std::operator<<<std::char_traits<char>>(v8, " is a leap year.", v9);
  std::ostream::operator<<(v10, &std::endl<char,std::char_traits<char>>);
}
else
{
  if ( (signed int)v15 % 400 )
  {
    v6 = std::ostream::operator<<(&std::cout, v15);
    v5 = std::operator<<<std::char_traits<char>>(v6, " is not a leap year.", v7);
  }
  else
  {
    v3 = std::ostream::operator<<(&std::cout, v15);
    v5 = std::operator<<<std::char_traits<char>>(v3, " is a leap year.", v4);
  }
  std::ostream::operator<<(v5, &std::endl<char,std::char_traits<char>>);
}
return 0;
```

- Bogus後(只截圖重點部份)：

```
31  std::operator<<<std::char_traits<char>>(&std::cout, "Enter a year: ", envp);
32  std::istream::operator>>(&std::cin, &v28);
33  if ( (signed int)v28 % 4 )
34  {
35    if ( y_4 >= 10 && (((_BYTE)x_3 - 1) * (_BYTE)x_3 & 1) != 0 )
36      goto LABEL_21;
37    while ( 1 )
38    {
39      v12 = std::ostream::operator<<(&std::cout, v28);
40      v14 = std::operator<<<std::char_traits<char>>(v12, " is not a leap year.", v13);
41      std::ostream::operator<<(v14, &std::endl<char,std::char_traits<char>>);
42      if ( y_4 < 10 || (((_BYTE)x_3 - 1) * (_BYTE)x_3 & 1) == 0 )
43        break;
44LABEL_21:
45      v25 = std::ostream::operator<<(&std::cout, v28);
46      v27 = std::operator<<<std::char_traits<char>>(v25, " is not a leap year.", v26);
47      std::ostream::operator<<(v27, &std::endl<char,std::char_traits<char>>);
48    }
49  }
50  else if ( (signed int)v28 % 100 )
51  {
52    if ( y_4 >= 10 && (((_BYTE)x_3 - 1) * (_BYTE)x_3 & 1) != 0 )
53      goto LABEL_20;
54    while ( 1 )
55    {
56      v9 = std::ostream::operator<<(&std::cout, v28);
57      v11 = std::operator<<<std::char_traits<char>>(v9, " is a leap year.", v10);
58      std::ostream::operator<<(v11, &std::endl<char,std::char_traits<char>>);
59      if ( y_4 < 10 || (((_BYTE)x_3 - 1) * (_BYTE)x_3 & 1) == 0 )
60        break;
61LABEL_20:
62      v22 = std::ostream::operator<<(&std::cout, v28);
63      v24 = std::operator<<<std::char_traits<char>>(v22, " is a leap year.", v23);
64      std::ostream::operator<<(v24, &std::endl<char,std::char_traits<char>>);
65    }
66  }
67  else
68  {
69    if ( (signed int)v28 % 400 )
70    {
71      if ( y_4 >= 10 && (((_BYTE)x_3 - 1) * (_BYTE)x_3 & 1) != 0 )
72        goto LABEL_18;
73      while ( 1 )
74      {
75        v6 = std::ostream::operator<<(&std::cout, v28);
76        v8 = std::operator<<<std::char_traits<char>>(v6, " is not a leap year.", v7);
77        std::ostream::operator<<(v8, &std::endl<char,std::char_traits<char>>);
78        if ( y_4 < 10 || (((_BYTE)x_3 - 1) * (_BYTE)x_3 & 1) == 0 )
79          break;
80LABEL_18:
81        v19 = std::ostream::operator<<(&std::cout, v28);
82        v21 = std::operator<<<std::char_traits<char>>(v19, " is not a leap year.", v20);
83        std::ostream::operator<<(v21, &std::endl<char,std::char_traits<char>>);
84      }
85    }
86    else
87    {
88      if ( y_4 >= 10 && (((_BYTE)x_3 - 1) * (_BYTE)x_3 & 1) != 0 )
89        goto LABEL_17;
90      while ( 1 )
91      {
92        v3 = std::ostream::operator<<(&std::cout, v28);
93        v5 = std::operator<<<std::char_traits<char>>(v3, " is a leap year.", v4);
94        std::ostream::operator<<(v5, &std::endl<char,std::char_traits<char>>);
95        if ( y_4 < 10 || (((_BYTE)x_3 - 1) * (_BYTE)x_3 & 1) == 0 )
```

> 可以發現原本只有三個運算式，變成了一堆運算式，輸出結果也不僅僅只有四個(print是否是leap year)而已，並且發現到有使用goto的語句。