BÁO CÁO THỰC HÀNH

Môn học: Lập trình hệ thống (NT209)

Lab 4 – Kỹ thuật dịch ngược – Nâng cao

GVHD: Đỗ Thị Thu Hiền

1. THÔNG TIN CHUNG:

(Liệt kê tất cả các thành viên trong nhóm)

Lớp: NT209.N21.ANTN.1

STT	Họ và tên	MSSV	Email
1	Nguyễn Nhật Quân	21522497	21522497@gm.uit.edu.vn
2	Phạm Nguyễn Hải Anh	21520586	21520586@gm.uit.edu.vn

2. NỘI DUNG THỰC HIỆN:[1]

STT	Công việc	Kết quả tự đánh giá
1	Pha 1	100%
2	Pha 2	100%
3	Pha 3	100%
4	Pha 4	100%
5	Pha 5	100%
6	Pha 6	100%
7	Pha bí mật (bonus)	100%

Phần bên dưới của báo cáo này là tài liệu báo cáo chi tiết của nhóm thực hiện.

[1] Ghi nội dung công việc, yêu cầu trong bài Thực hành

BÁO CÁO CHI TIẾT

Phương pháp phân tích: Phân tích tĩnh/Remote Debug (chụp hình ảnh minh chứng)

```
28  initialize_bomb();
29  puts("Welcome to my fiendish little bomb. You have 6 phases with");
30  puts("which to blow yourself up. Have a nice day!");
31  v3 = read_line();
32  phase_1(v3);
33  phase_defused();
34  puts("Phase 1 defused. How about the next one?");
```

Bắt đầu trước khi vào pha 1 thì sẽ hiện ra 2 câu chào mừng

Hàm để nổ bomb:

```
1 void __noreturn explode_bomb()
2 {
3   puts("\nBOOM!!!");
4   puts("The bomb has blown up.");
5   exit(8);
6 }
```

Nếu sai input ở pha nào thì bomb sẽ nổ và sẽ thoát chương trình

1. Pha 1

+ Đoạn mã/ Hàm xử lý:

```
1 int __cdecl phase_1(int a1)
2 {
3   int result; // eax@1
4
5   result = strings_not_equal(a1, "I am the mayor. I can do anything I want.");
6   if ( result )
7     explode_bomb();
8   return result;
9 }
```

Tham số đầu vào của phase_1 là kết quả của hàm read_line() (nhập từ bàn phím) được lưu vào biến v3

+ Yêu cầu đầu vào: không có yêu cầu đầu vào => là một chuỗi ký tự bất kì

+ Kiểm tra đầu vào: được thực hiện bởi hàm strings_not_equal

```
signed int __cdecl strings_not_equal(int a1, int a2)
{
  int v2; // ebx@1
  signed int result; // eax@2
  int v4; // [sp+8h] [bp-Ch]@3
  int v5; // [sp+Ch] [bp-8h]@3

  v2 = string_length(a1);
  if ( v2 == string_length(a2) )
  {
    v4 = a1;
    v5 = a2;
    while ( *(_BYTE *)v4 )
    {
      if ( *(_BYTE *)v4 != *(_BYTE *)v5 )
        return 1;
      ++v4;
      ++v5;
    }
    result = 0;
  }
  else
  {
    result = 1;
  }
  return result;
}
```

- Hàm này sẽ có hai tham số đầu vào: input và một chuỗi cho trước
- Đầu tiên sẽ kiểm tra độ dài của hai chuỗi nếu khác nhau thì trả về 1

```
int __cdecl string_length(int a1)
{
  int v2; // [sp+8h] [bp-8h]@1
  int v3; // [sp+Ch] [bp-4h]@1

  v3 = a1;
  v2 = 0;
  while ( *(_BYTE *)v3 )
  {
    ++v3;
    ++v2;
  }
  return v2;
}
```

Đây là hàm để tính độ dài chuỗi

- Nếu giống nhau thì sẽ kiểm tra từng kí tự, nếu không có kí tự nào khác nhau thì trả về 0, còn nếu có ít nhất 1 kí tự khác nhau thì trả về 1
- Kết quả trả về được lưu vào biến result

```
if ( result )
  explode_bomb();
return result;
```

- Nếu result bằng 1 thì sẽ nổ bomb (hai chuỗi khác nhau)

+ input phải giống với chuỗi cho trước: "I am the mayor. I can do anything I want."\

+ Kết quả:

```
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
I am the mayor. I can do anything I want.
Phase 1 defused. How about the next one?
```

2. Pha 2

+ Đoạn mã/ Hàm xử lý:

```
1 int __cdecl phase_2(int a1)
2 {
3   signed int i; // [sp+10h] [bp-28h]@4
4   int v3; // [sp+14h] [bp-24h]@1
5   int v4; // [sp+18h] [bp-20h]@2
6   int v5; // [sp+2Ch] [bp-Ch]@1
7
8   v5 = *MK_FP(__GS__, 20);
9   read_six_numbers(a1, &v3);
10  if ( v3 || v4 != 1 )
11    explode_bomb();
12  for ( i = 2; i <= 5; ++i )
13  {
14    if ( *(&v3 + i) != *(&v3 + i - 2) + *(&v3 + i - 1) )
15      explode_bomb();
16  }
17  return *MK_FP(__GS__, 20) ^ v5;
18 }
```

Tham số truyền vào phase_2 là chuỗi được nhập từ bàn phím

+ Yêu cầu đầu vào: nhập vào một chuỗi với format “a b c d e f” với a -> f là các số tự nhiên. Yêu cầu này là điều kiện cần để tiếp tục thực hiện kiểm tra (tuy nhiên chúng ta có thể nhập dư ra mà vẫn kiểm tra đúng và không nổ bomb)

```
result = __isoc99_sscanf(a1, "%d %d %d %d %d %d", a2, a2 + 4, a2 + 8, a2 + 12, a2 + 16, a2 + 20);
```

Format được lấy từ hàm read_six_number

+ Kiểm tra đầu vào:

- Input sẽ đi qua hàm read_six_number

```
1 int __cdecl read_six_numbers(int a1, int a2)
2 {
3   int result; // eax@1
4
5   result = __isoc99_sscanf(a1, "%d %d %d %d %d %d", a2, a2 + 4, a2 + 8, a2 + 12, a2 + 16, a2 + 20);
6   if ( result <= 5 )
7     explode_bomb();
8   return result;
9 }
```

- Hàm này sẽ tách ra 6 ký tự đầu theo như format bên trên ra thành số integer và lưu từng số vào mảng v3 ở phase_2 bên ngoài

```
if ( v3 || v4 != 1 )
  explode_bomb();
```

- v3 và v4 lần lượt là hai phần tử đầu tiên của mảng v3 trên hay là 2 số đầu tiên nhập vào => 2 số này lần lượt là 0 và 1 nếu không bomb sẽ nổ

```
12  for ( i = 2; i <= 5; ++i )
13  {
14    if ( *(&v3 + i) != *(&v3 + i - 2) + *(&v3 + i - 1) )
15      explode_bomb();
16  }
17  return *MK_FP(__GS__, 20) ^ v5;
18}
```

- Tiếp đến là kiểm tra các phần tử trong mảng với điều kiện như sau:
  v3[i] = v3[i-2] + v3[i-1] (i: 2->5)
- Có thể nói là số ở sau bằng tổng 2 số liền trước (fibonanci)
- Sau khi kiểm tra đúng rồi thì phase_2 sẽ kết thúc mà không bị nổ

+ Nhận xét về input: input là một chuỗi fibonanci với 2 số đầu tiên là 0 và 1

  ⇨ Input: “0 1 1 2 3 5”
  ⇨ Tuy nhiên ta có thể nhập dư ra như thế này: "0 1 1 2 3 5abc” hoặc ”0 1 1 2 3 5 123” (điều kiện chỉ là 6 số đầu)

+ Kết quả:

```
Phase 1 defused. How about the next one?
0 1 1 2 3 5
That's number 2.  Keep going!
```

```
Phase 1 defused. How about the next one?
0 1 1 2 3 5abc
That's number 2.  Keep going!
```

3. Pha 3

+ Đoạn mã/ Hàm xử lý:

```
int __cdecl phase_3(int a1)
{
  int result; // eax@23
  char v2; // [sp+1Eh] [bp-1Ah]@1
  char v3; // [sp+1Fh] [bp-19h]@4
  int v4; // [sp+20h] [bp-18h]@1
  int v5; // [sp+24h] [bp-14h]@1
  int v6; // [sp+28h] [bp-10h]@1
  int v7; // [sp+2Ch] [bp-Ch]@1

  v7 = *MK_FP(__GS__, 20);
  v6 = 0;
  v6 = __isoc99_sscanf(a1, "%d %c %d", &v4, &v2, &v5);
  if ( v6 <= 2 )
    explode_bomb();
  switch ( v4 )
  {
    case 0:
      v3 = 118;
      if ( v5 != 335 )
        explode_bomb();
      return result;
    case 1:
      v3 = 97;
      if ( v5 != 465 )
        explode_bomb();
      return result;
```

```
      if ( v5 != 465 )
        explode_bomb();
      return result;
    case 2:
      v3 = 111;
      if ( v5 != 238 )
        explode_bomb();
      return result;
    case 3:
      v3 = 116;
      if ( v5 != 786 )
        explode_bomb();
      return result;
    case 4:
      v3 = 106;
      if ( v5 != 81 )
        explode_bomb();
      return result;
    case 5:
      v3 = 114;
      if ( v5 != 661 )
        explode_bomb();
      return result;
    case 6:
      v3 = 104;
      if ( v5 != 736 )
        explode_bomb();
```

```
48    case 6:
49      v3 = 104;
50      if ( v5 != 736 )
51        explode_bomb();
52      return result;
53    case 7:
54      v3 = 111;
55      if ( v5 != 969 )
56        explode_bomb();
57      return result;
58    default:
59      v3 = 116;
60      explode_bomb();
61      return result;
62  }
63  if ( v3 != v2 )
64    explode_bomb();
65  return *MK_FP(__GS__, 20) ^ v7;
66}
```

+ Yêu cầu đầu vào: Đầu vào phải đi theo format "(số) (kí tự) (số)"

```
v6 = __isoc99_sscanf(a1, "%d %c %d", &v4, &v2, &v5);
```

+ Kiểm tra đầu vào:

```
16  switch ( v4 )
17  {
18    case 0:
19      v3 = 118;
```

- Đầu tiên sẽ kiểm tra số đầu tiên nếu bằng bao nhiêu thì sẽ đi vào case phù hợp
- Chúng ta sẽ phân tích với với case = 0

```
18    case 0:
19      v3 = 118;
20      if ( v5 != 335 )
21        explode_bomb();
22      return result;
```

- Với case bằng 0, ký tự phải = 118 ('v' theo ascii), số thứ 2 phải = 335
- Nếu đúng như vậy thì sẽ phá được phase_3

+ Nhận xét input: input sẽ được chia theo từng case của v4 (số đầu tiên nhập vào). Tiếp tục như vậy ta sẽ có được các input hợp lệ khác

⇨ 0 v 335
⇨ 1 a 465
⇨ 2 o 238
⇨ 3 t 786
⇨ 4 j 81
⇨ 5 r 661

⇨ 6 h 736

⇨ 7 o 969

⇨ Tuy nhiên cũng giống như phase 2 ta có thể nhập thừa input ở cuối chuỗi (chỉ cần ký tự liền kề khác số) do tính chất của sscanf và %d ở cuối

+ Kết quả:

```
That's number 2.  Keep going!
0 v 335
Halfway there!
```

4. Pha 4

+ Đoãn mã/ Hàm xử lý:

```
1 int __cdecl phase_4(int a1)
2 {
3   int v2; // [sp+18h] [bp-20h]@1
4   int v3; // [sp+1Ch] [bp-1Ch]@1
5   int v4; // [sp+20h] [bp-18h]@1
6   int v5; // [sp+24h] [bp-14h]@5
7   int v6; // [sp+28h] [bp-10h]@5
8   int v7; // [sp+2Ch] [bp-Ch]@1
9
10  v7 = *MK_FP(__GS__, 20);
11  v4 = __isoc99_sscanf(a1, "%d %d", &v2, &v3);
12  if ( v4 != 2 || v2 < 0 || v2 > 14 )
13    explode_bomb();
14  v5 = 0;
15  v6 = func4(v2, 0, 14);
16  if ( v6 != v5 || v3 != v5 )
17    explode_bomb();
18  return *MK_FP(__GS__, 20) ^ v7;
19 }
```

+ Yêu cầu đầu vào: Đầu vào phải theo format "(số) (số)"

+ Kiểm tra đầu vào:

- 2 số lần lượt được lưu vào v2 và v3
- Điều kiện đầu tiên là phải truyền thành công 2 số vào v2 và v3; v2 trong khoảng 0 đến 14
- Sau khi đưa v2 vào một hàm func4(v2, 0, 14) thực hiện chức năng tính toán thì có kết quả trả về được gán vào v6

```
int __cdecl func4(int a1, int a2, int a3)
{
  int result; // eax@2
  int v4; // [sp+Ch] [bp-Ch]@1

  v4 = (a3 - a2) / 2 + a2;
  if ( v4 <= a1 )
  {
    if ( v4 >= a1 )
      result = 0;
    else
      result = 2 * func4(a1, v4 + 1, a3) + 1;
  }
  else
  {
    result = 2 * func4(a1, a2, v4 - 1);
  }
  return result;
}
```

- Điều kiện để bomb không nổ là v6 == 0 và v3 == 0

+ Nhận xét input:

- Vì vậy chỉ cần viết ra một đoạn code bằng ngôn ngữ lập trình để chạy func4 với v2 chạy từ 0 đến 14

```
#include <stdio.h>
#include <string.h>

int __cdecl func4(int a1, int a2, int a3)
{
  int result; // eax@2
  int v4; // [sp+Ch] [bp-Ch]@1

  v4 = (a3 - a2) / 2 + a2;
  if ( v4 <= a1 )
  {
    if ( v4 >= a1 )
      result = 0;
    else
      result = 2 * func4(a1, v4 + 1, a3) + 1;
  }
  else
  {
    result = 2 * func4(a1, a2, v4 - 1);
  }
  return result;
}


int main(int argc, char **argv) {
    for (int i = 0; i <= 14; i++) {
        printf("%d: %d\n", i, func4(i, 0, 14));
    }
}
```

```
0: 0
1: 0
2: 4
3: 0
4: 2
5: 2
6: 6
7: 0
8: 1
9: 1
10: 5
11: 1
12: 3
13: 3
14: 7
```

- Chúng ta chỉ cần chọn ra những kết quả mà ở sau dấu ":" bằng 0: 0, 1, 3, 7. Từ đó kết hợp với v3 = 0 thì được kết quả input
  - ⇨ 0 0
  - ⇨ 1 0
  - ⇨ 3 0
  - ⇨ 7 0
  - ⇨ Cũng giống như pha 2 và 3 thì chúng ta có thể dựa vào format %d để nhập dư ra ở cuối với ký tự liền kề sau %d cuối cùng phải khác số

+ Kết quả:

```
Halfway there!
0 0
So you got that one.  Try this one.
```

5. Pha 5

+ Đoạn mã/ Hàm xử lý:

```
1  int __cdecl phase_5(int a1)
2  {
3    signed int i; // [sp+1Ch] [bp-1Ch]@3
4    char v3[6]; // [sp+25h] [bp-13h]@4
5    char v4; // [sp+2Bh] [bp-Dh]@6
6    int v5; // [sp+2Ch] [bp-Ch]@1
7
8    v5 = *MK_FP(__GS__, 20);
9    if ( string_length(a1) != 6 )
10     explode_bomb();
11   for ( i = 0; i <= 5; ++i )
12     v3[i] = array_2706[*(_BYTE *)(i + a1) & 0xF];
13   v4 = 0;
14   if ( strings_not_equal(v3, "sabres") )
15     explode_bomb();
16   return *MK_FP(__GS__, 20) ^ v5;
17 }
```

+ Điều kiện đầu vào: một chuỗi gồm 6 ký tự

```
if ( string_length(a1) != 6 )
  explode_bomb();
```

+ Kiểm tra input:

- Lần lượt gán vào một chuỗi v3 các kí tự dựa trên một chuỗi array_2706 được lấy index từ input[i] (giá trị integer tương ứng với mỗi ký tự từ input theo ascii, i chạy từ 0 -> 5) AND 0xF

```
for ( i = 0; i <= 5; ++i )
  v3[i] = array_2706[*(_BYTE *)(i + a1) & 0xF];
```

```
.data:0804D1A8 array_2706      db 'm'                  ; DATA XREF: phase_5+4D↑r
.data:0804D1A9 aAduiersnfotvby db 'aduiersnfotvbyl',0
.data:0804D1B9                 align 10h
```

- array_2706 có thể được viết lại như 'maduiersnfotvbyl'
- Sau đó đem v3 so sánh với chuỗi "sabres" nếu không có gì khác nhau thì phase 5 sẽ được phá

+ Nhận xét input:

- Do v3 phải giống "sabres" nên có thể suy ra được index cần thiết từ 'maduiersnfotvbyl' lần lượt là 7, 1, 13, 6, 5, 7
- Hay nói cách khác input[i] & 0xF = 7, 1, 13, 6, 5, 7
- Từ đó có thể viết code ra để cho chương trình tự chạy

```python
array = [7, 1, 13, 6, 5, 7]

s0 = ""
s1 = ""
s2 = ""
s3 = ""
s4 = ""
s5 = ""

for i in range(len(array)):
    var_name = "s{}".format(i)
    locals()[var_name] = ""
    char = 33
    while char < 127:
        if (char & 15 == array[i]):
            locals()[var_name] += chr(char)
        char += 1

for i0 in s0:
    for i1 in s1:
        for i2 in s2:
            for i3 in s3:
                for i4 in s4:
                    for i5 in s5:
                        with open("output.txt", "a") as file:
                            file.write(i0 + i1 + i2 + i3 + i4 + i5 + '\n')
```

- Kết quả chạy được sẽ lưu vào file output.txt
- Ta có kết quả dạng như thế này

```
'!-&%'
'!-&%7
'!-&%G
'!-&%W
'!-&%g
'!-&%w
```

```
wq}veg
wq}vew
wq}vu'
wq}vu7
wq}vuG
wq}vuW
wq}vug
wq}vuw
```

- Có tổng cộng 46656 input hợp lệ cho phase này

+ kết quả:

```
So you got that one.  Try this one.
wq}vuw
Good work!  On to the next...
```

6. Pha 6

+ Đoạn mã/ Hàm xử lý:

```
int __cdecl phase_6(int a1)
{
  int v2; // [sp+1Ch] [bp-4Ch]@13
  int v3; // [sp+1Ch] [bp-4Ch]@18
  int v4; // [sp+1Ch] [bp-4Ch]@21
  signed int i; // [sp+20h] [bp-48h]@1
  signed int k; // [sp+20h] [bp-48h]@12
  signed int m; // [sp+20h] [bp-48h]@18
  signed int n; // [sp+20h] [bp-48h]@21
  int j; // [sp+24h] [bp-44h]@5
  int l; // [sp+24h] [bp-44h]@13
  int v11; // [sp+28h] [bp-40h]@18
  int v12[6]; // [sp+2Ch] [bp-3Ch]@1
  int v13[6]; // [sp+44h] [bp-24h]@16
  int v14; // [sp+5Ch] [bp-Ch]@1

  v14 = *MK_FP(__GS__, 20);
  read_six_numbers(a1, v12);
  for ( i = 0; i <= 5; ++i )
  {
    if ( v12[i] <= 0 || v12[i] > 6 )
      explode_bomb();
    for ( j = i + 1; j <= 5; ++j )
    {
      if ( v12[i] == v12[j] )
        explode_bomb();
    }
```

```
        explode_bomb();
    }
  }
  for ( k = 0; k <= 5; ++k )
  {
    v2 = (int)&node1;
    for ( l = 1; v12[k] > l; ++l )
      v2 = *(_DWORD *)(v2 + 8);
    v13[k] = v2;
  }
  v11 = v13[0];
  v3 = v13[0];
  for ( m = 1; m <= 5; ++m )
  {
    *(_DWORD *)(v3 + 8) = v13[m];
    v3 = *(_DWORD *)(v3 + 8);
  }
  *(_DWORD *)(v3 + 8) = 0;
  v4 = v11;
  for ( n = 0; n <= 4; ++n )
  {
    if ( *(_DWORD *)v4 < **(_DWORD **)(v4 + 8) )
      explode_bomb();
    v4 = *(_DWORD *)(v4 + 8);
  }
  return *MK_FP(__GS__, 20) ^ v14;
}
```

+ Điều kiện đầu vào: gồm 6 số cách nhau bởi dấu cách kiểu như "a b c d e f"

```
1 int __cdecl read_six_numbers(int a1, int a2)
2 {
3   int result; // eax@1
4
5   result = __isoc99_sscanf(a1, "%d %d %d %d %d %d", a2, a2 + 4, a2 + 8, a2 + 12, a2 + 16, a2 + 20);
6   if ( result <= 5 )
7     explode_bomb();
8   return result;
9 }
```

+ Kiểm tra input:

```
19   for ( i = 0; i <= 5; ++i )
20   {
21     if ( v12[i] <= 0 || v12[i] > 6 )
22       explode_bomb();
23     for ( j = i + 1; j <= 5; ++j )
24     {
25       if ( v12[i] == v12[j] )
26         explode_bomb();
27     }
28   }
```

- Đầu tiên sau khi lấy được 6 số và lưu vào mảng v12, chương trình sẽ kiểm tra để chắc chắn các số ở trong khoảng 1 -> 6 và không có số nào trùng nhau

```
29   for ( k = 0; k <= 5; ++k )
30   {
31     v2 = (int)&node1;
32     for ( l = 1; v12[k] > l; ++l )
33       v2 = *(_DWORD *)(v2 + 8);
34     v13[k] = v2;
35   }
```

- Tiếp đến sẽ gắn địa chỉ của các node vào từng phần tử của v13, thứ tự của node sẽ phụ thuộc vào input. VD:
  - Input: 2 3 5 6 1 4
  - v13[0] = &node2, v13[1] = &node3, v13[2] = &node5, v13[3] = &node6, v13[4] = &node1, v13[1] = &node4
- Các node là một linked list được tạo sẵn trong chương trình

```
.data:0804D0E8 node1           db 0BCh ; +
.data:0804D0E9                 db    3
.data:0804D0EA                 db    0
.data:0804D0EB                 db    0
.data:0804D0EC                 db    1
.data:0804D0ED                 db    0
.data:0804D0EE                 db    0
.data:0804D0EF                 db    0
.data:0804D0F0                 db 0DCh ; _
.data:0804D0F1                 db 0D0h ; -
.data:0804D0F2                 db    4
.data:0804D0F3                 db    8
```

  - Phần trên là dữ liệu, 4 byte cuối sẽ trỏ đến node tiếp theo:

node1->node2->node3->node4->node5->node6

```
v11 = v13[0];
v3 = v13[0];
```

- Gắn địa chỉ của node thứ a (theo input) cho v11 và v3

```
38  for ( m = 1; m <= 5; ++m )
39  {
40    *(_DWORD *)(v3 + 8) = v13[m];
41    v3 = *(_DWORD *)(v3 + 8);
42  }
43  *(_DWORD *)(v3 + 8) = 0;
44  v4 = v11;
```

- Vòng lặp này là sắp xếp lại thứ tự các node trỏ tới nhau theo input thay vì trỏ từ 1 đến 6 như mặc định. VD:
  - Input: 2 3 5 6 1 4
  - node2-> node3-> node5-> node6-> node1-> node4
  - node cuối trỏ vào null

```
45  for ( n = 0; n <= 4; ++n )
46  {
47    if ( *(_DWORD *)v4 < **(_DWORD **)(v4 + 8) )
48      explode_bomb();
49    v4 = *(_DWORD *)(v4 + 8);
50  }
51  return *MK_FP(__GS__, 20) ^ v14;
52}
```

- Đây là phần cuối cùng, nó kiểm tra data trong node trước có lớn hơn node sau không, nếu thỏa mãn tất cả thì sẽ pass phase_6

+ Nhận xét input:

- Như điều kiện ở trên, ta chỉ cần tìm và sắp xếp các node theo thứ tự độ lớn data giảm dần
  - Node1 = 0x 00 00 03 BC
  - Node2 = 0x 00 00 00 FE
  - Node5 = 0x 00 00 00 F5
  - Node6 = 0x 00 00 00 E6
  - Node3 = 0x 00 00 00 D4
  - Node4 = 0x 00 00 00 3A

  ⇨ Input hợp lệ: 1 2 5 6 3 4

  ⇨ Dựa vào tính chất của %d ta cũng có thể nhập thừa ra ở cuối như những phase trên

+ Kết quả:

```
Good work!  On to the next...
1 2 5 6 3 4
Congratulations! You've defused the bomb!
```

7. Pha bí mật (Bonus)

+ Cách để vào pha bí mật:

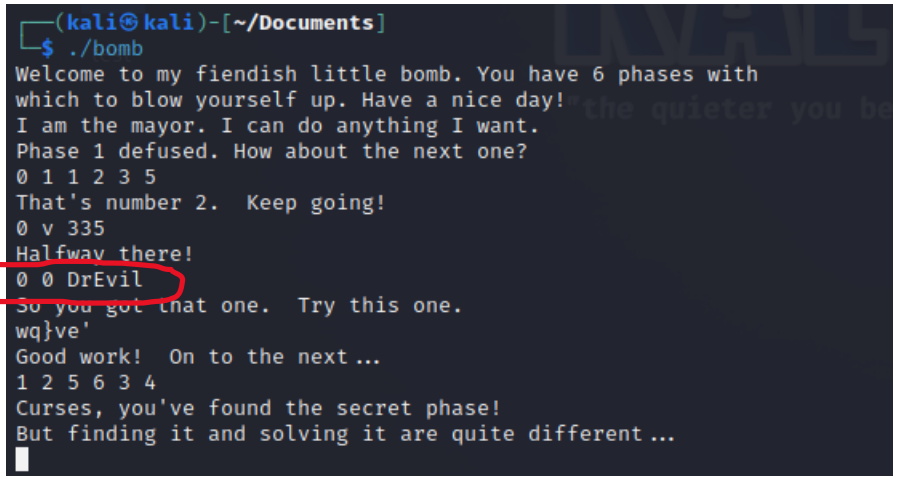
```
1  int phase_defused()
2  {
3    char v1; // [sp+0h] [bp-68h]@2
4    char v2; // [sp+4h] [bp-64h]@2
5    int v3; // [sp+8h] [bp-60h]@2
6    char v4; // [sp+Ch] [bp-5Ch]@2
7    int v5; // [sp+5Ch] [bp-Ch]@1
8
9    v5 = *MK_FP(__GS__, 20);
10   if ( num_input_strings == 6 )
11   {
12     v3 = __isoc99_sscanf(&unk_804D4F0, "%d %d %s", &v1, &v2, &v4);
13     if ( v3 == 3 && !strings_not_equal((int)&v4, (int)"DrEvil") )
14     {
15       puts("Curses, you've found the secret phase!");
16       puts("But finding it and solving it are quite different...");
17       secret_phase();
18     }
19     puts("Congratulations! You've defused the bomb!");
20   }
21   return *MK_FP(__GS__, 20) ^ v5;
22 }
```

- Quan sát trong hàm phase_defused thì ta thấy có gọi ra secret_phase()
- Tuy nhiên cần phải có điều kiện thì mới gọi được:
    - Num_input_strings (số input nhập vào hay nói cách khác là số phase gỡ được trước đó) phải bằng 6
    - Chúng ta sẽ đi xem qua hàm nhập input read_line()

```
36   *(_BYTE *)(v4 - 1 + 80 * num_input_strings + 134534144) = 0;
37   v2 = num_input_strings++;
38   return 80 * v2 + 134534144;
39 }
```

    - Hàm read_line này trả về một địa chỉ, có nghĩa là mỗi input của một phase sẽ được nằm ở trong một địa chỉ nào đó
    - Sau khi dò qua và tính toán qua các phase thì em thấy ở lần nhập phase_4, v2 = 3, thì địa chỉ trả về bằng với địa chỉ của biến unk_804D4F0

- Điều này có nghĩa là input ở phase_4 sẽ liên quan đến phase_secret
- Dựa trên tính chất nhập thừa nêu trên thì chúng ta sẽ nhập đúng input của phase_4 và thêm "DrEvil" để đi vào phase_secret



+ Đoạn mã/ Hàm xử lý:

```
1 int secret_phase()
2 {
3   char *nptr; // ST14_4@1
4   signed int v2; // [sp+8h] [bp-10h]@1
5
6   nptr = (char *)read_line();
7   v2 = atoi(nptr);
8   if ( v2 <= 0 || v2 > 1001 )
9     explode_bomb();
10  if ( fun7((int)&n1, v2) != 3 )
11    explode_bomb();
12  puts("Wow! You've defused the secret stage!");
13  return phase_defused();
14 }
```

+ Điều kiện input: input là một chuỗi ký tự số sau đó sẽ được chuyển sang dạng integer.

+ Kiểm tra input:

- Sau khi chuyển sang integer được gán vào v2, kiểm tra xem số đó có nằm trong khoảng 1->1001 không, nếu không bomb sẽ nổ
- Sau đó đưa địa chỉ n1 (một địa chỉ nào đó trong chương trình) và v2 vào hàm fun7

```
1  int __cdecl fun7(int a1, int a2)
2  {
3    int result; // eax@2
4
5    if ( a1 )
6    {
7      if ( *(_DWORD *)a1 <= a2 )
8      {
9        if ( *(_DWORD *)a1 == a2 )
10         result = 0;
11       else
12         result = 2 * fun7(*(_DWORD *)(a1 + 8), a2) + 1;
13     }
14     else
15     {
16       result = 2 * fun7(*(_DWORD *)(a1 + 4), a2);
17     }
18   }
19   else
20   {
21     result = -1;
22   }
23   return result;
24 }
```

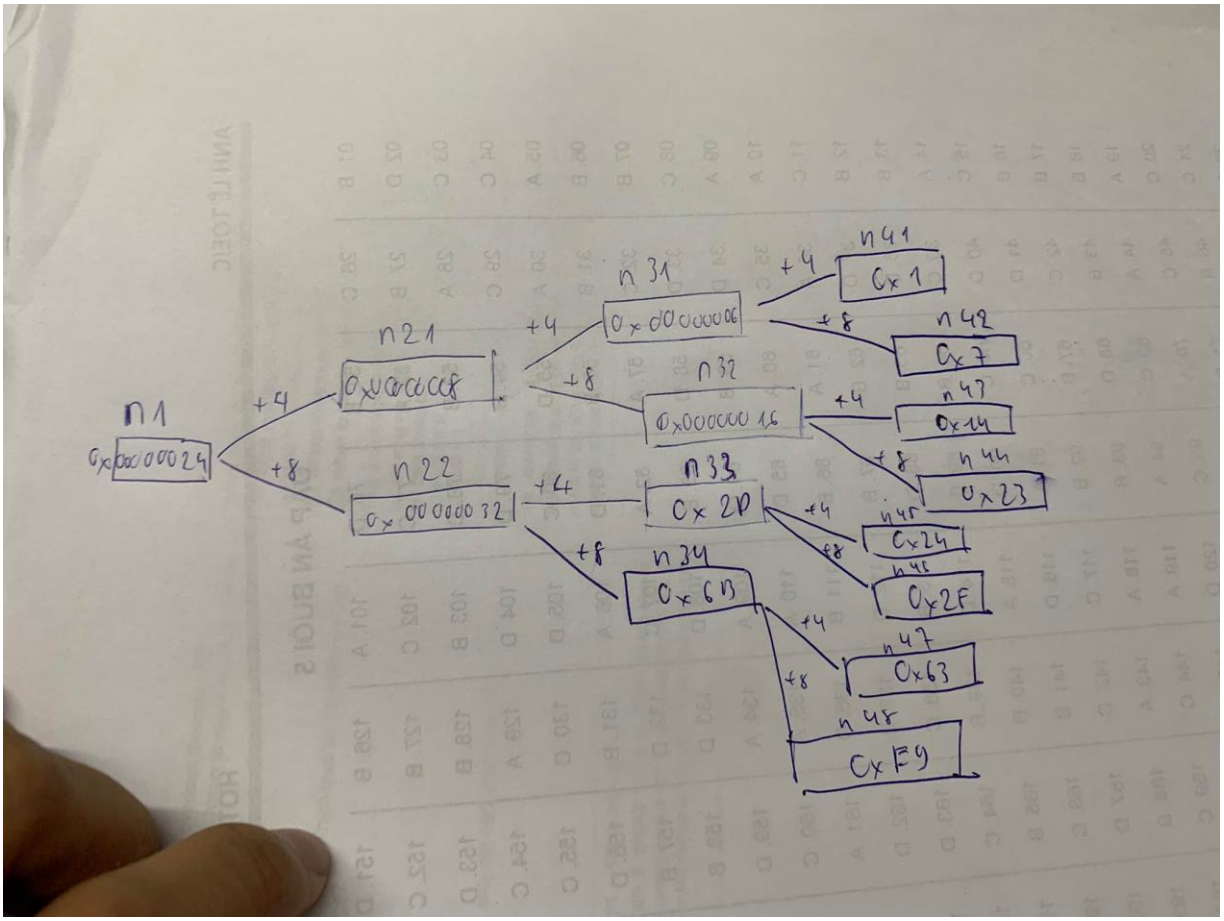
- Nếu kết quả trả về của hàm fun7 mà bằng 3 thì bomb sẽ không nổ và hoàn thành các phase

+ Nhận xét input:

- Ta có thể thấy bước quan trọng nhất là tính toán và trả về kết quả trong fun7
- Trước tiên thì cần phải xem n1 là gì

```
.data:0804D19C                  public n1
.data:0804D19C n1               db  24h ; $
.data:0804D19D                  db    0
.data:0804D19E                  db    0
.data:0804D19F                  db    0
.data:0804D1A0                  db  90h ; É
.data:0804D1A1                  db 0D1h ; -
.data:0804D1A2                  db    4
.data:0804D1A3                  db    8
.data:0804D1A4                  db  84h ; ä
.data:0804D1A5                  db 0D1h ; -
.data:0804D1A6                  db    4
.data:0804D1A7                  db    8
```

- n1 có thể được xem như 1 node với phần trên là data, bên dưới là địa chỉ của 2 node khác, nếu cứ tiếp tục như vậy ta sẽ có 1 node chỉa ra 2 node khác, đây chính là binary tree



- Em ghi lại như vậy cho dễ quan sát
- Hàm fun7 là một hàm đệ quy cho binary tree, từ đó em viết lại bằng python để tìm ra input như sau

```python
# Define the binary tree node class
class Node:
    def __init__(self, value=None, left=None, right=None):
        self.value = value
        self.left = left
        self.right = right

# Initialize the binary tree with some values
root = Node(0x24)

root.left = Node(0x32)
root.right = Node(0x08)

root.left.left = Node(0x6B)
root.left.right = Node(0x2D)
root.right.left = Node(0x16)
root.right.right = Node(0x06)

root.left.left.left = Node(0xE9)
root.left.left.right = Node(0x63)
root.left.right.left = Node(0x2F)
root.left.right.right = Node(0x24)
root.right.left.left = Node(0x23)
root.right.left.right = Node(0x14)
root.right.right.left = Node(0x07)
root.right.right.right = Node(0x01)
```

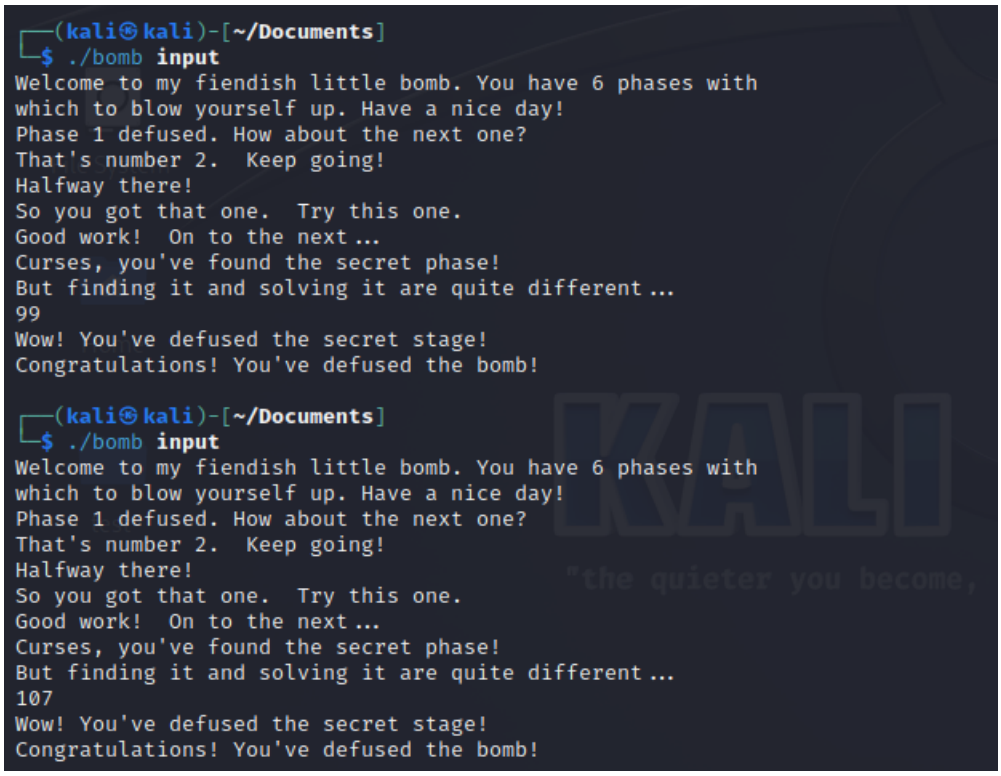
```python

def fun7(a1, a2):
    if a1 is not None:
        if a1.value <= a2:
            if a1.value == a2:
                result = 0
            else:
                result = 2 * fun7(a1.left, a2) + 1
        else:
            result = 2 * fun7(a1.right, a2)
    else:
        result = -1
    return result

i = 1
while i <= 1001:
    if fun7(root, i) == 3:
        print(i)
    i += 1
```

- Sau khi chạy code thì được 2 kết quả như sau: 99 và 107

+ Kết quả:



YÊU CẦU CHUNG

Báo cáo:

- File **.PDF**.
- Đặt tên theo định dạng: **[Mã lớp]-Lab4_NhomX_MSSV1-MSSV2.pdf** (trong đó X là số thứ tự nhóm, MSSV gồm đầy đủ MSSV của tất cả các thành viên thực hiện bài thực hành).

  Ví dụ: [NT209.N21.ANTN.1]-Lab4_Nhom2_21520001-21520013.pdf.
- Nộp file báo cáo trên theo thời gian đã thống nhất tại courses.uit.edu.vn.

Đánh giá:

- Hoàn thành tốt yêu cầu được giao.
- Có nội dung mở rộng, ứng dụng.

Bài sao chép, trễ, … sẽ được xử lý tùy mức độ vi phạm.

HẾT