

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.N22.ATCL.1

STT	Họ và tên	MSSV	Email
1	Võ Sỹ Minh	21521146	21521146@gm.uit.edu.vn
2	Lê Huy Hùng	21520888	21520888@gm.uit.edu.vn
3			

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)	

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)

1. Pha 1

Đoạn mã giả cung cấp bởi IDA:

```
int __cdecl phase_1(int a1)
{
    int result; // eax@1

    result = strings_not_equal(a1, "He is evil and fits easily into most overhead storage bins.");
    if ( result )
        explode_bomb();
    return result;
}
```

Ta thấy có một hàm “**strings_not_equal**” để so sánh a1 và một chuỗi có sẵn “**He is evil and fits easily into most overhead storage bins.**”

- Ta có thể biết đầu vào phải là một chuỗi ký tự

Nếu result có giá trị true hay chuỗi đầu vào không bằng chuỗi có sẵn trên thì sẽ nổ bomb

- Ta cần nhập chuỗi có sẵn ở trên hay input của pha 1 là “**He is evil and fits easily into most overhead storage bins.**”

Kiểm thử:



```
kali@kali: ~/Downloads
File Actions Edit View Help
(kali@kali)-[~/Downloads]
$ ./bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
He is evil and fits easily into most overhead storage bins.
Phase 1 defused. How about the next one?
█
```

2. Pha 2

Đoạn mã giả được cung cấp:

```
int __cdecl phase_2(int a1)
{
    signed int i; // [sp+10h] [bp-28h]@3
    int v3[6]; // [sp+14h] [bp-24h]@1
    int v4; // [sp+2Ch] [bp-Ch]@1

    v4 = *MK_FP(__GS__, 20);
    read_six_numbers(a1, v3);
    if ( v3[0] < 0 )
        explode_bomb();
    for ( i = 1; i <= 5; ++i )
    {
        if ( v3[i] != v3[i - 1] + i )
            explode_bomb();
    }
    return *MK_FP(__GS__, 20) ^ v4;
}
```

Hàm nhập “read_six_numbers” input của pha 2: có dạng 6 chữ số cách nhau bằng dấu cách

```
int __cdecl read_six_numbers(int a1, int a2)
{
    int result; // eax@1

    result = __isoc99_sscanf(a1, "%d %d %d %d %d %d", a2, a2 + 4, a2 + 8, a2 + 12, a2 + 16, a2 + 20);
    if ( result <= 5 )
        explode_bomb();
    return result;
}
```

Vậy ta thấy đầu vào là một dãy nhiều số nguyên cách nhau bằng dấu cách

Điều kiện để bắt đầu khởi tạo dãy số này thông qua số đầu tiên (ở đây là v3[0]) là v3[0] < 0 thì kích hoạt vomb

- Vì vậy số đầu tiên của dãy có thể là số nguyên bất kỳ > 0

Do ở đây có điều kiện v3[i] != v3[i-1] + i thì sẽ kích hoạt bomb nên ta biết được:

- Các số tiếp theo được hình thành bằng cách lấy chỉ số index hiện tại cộng với số đứng trước nó. (v3[i] = v3[i-1] + i)

Vì vậy ta có một dãy số đáp ứng như sau:

Số thứ nhất (v3[0]) : 1 (yếu tố khởi tạo dãy)

Số thứ hai (v3[1]): 1 + 1 = 2

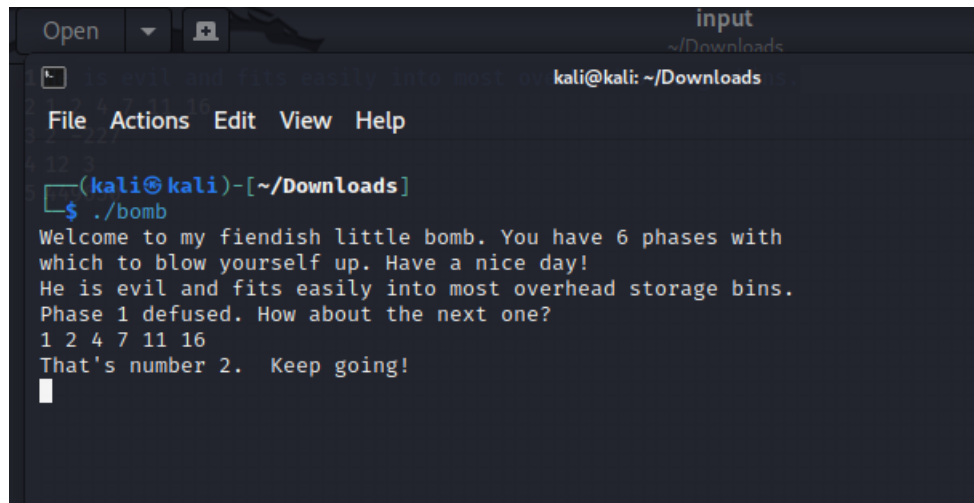
Số thứ ba (v3[2]): 2 + 2 = 4

Số thứ bốn(v3[3]): 4 + 3 = 7

Số thứ năm(v3[4]): 7 + 4 = 11

Số thứ sáu($v3[5]$): $11 + 5 = 16$

- dãy số $v3$ của trường hợp $v3[0] = 1$: 1 2 4 7 11 16



```

kali@kali: ~/Downloads
File Actions Edit View Help
(kali@kali)-[~/Downloads]
$ ./bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
He is evil and fits easily into most overhead storage bins.
Phase 1 defused. How about the next one?
1 2 4 7 11 16
That's number 2. Keep going!

```

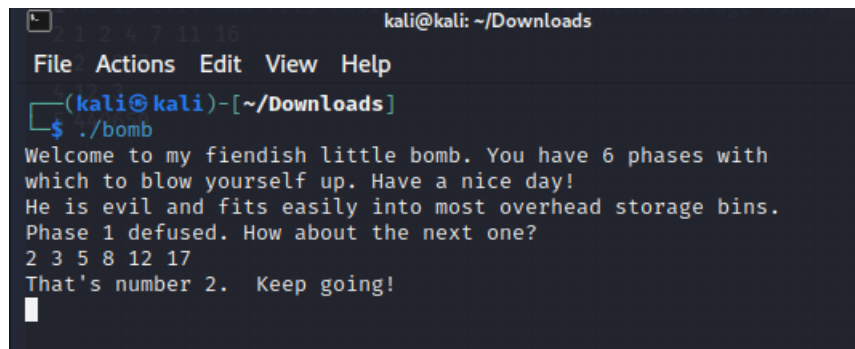
Vậy đầu vào của pha 2 có **vô hạn** trường hợp, miễn máy tính có thể xử lý được thì bất kỳ một số nguyên dương nào được chọn làm $v3[0]$ thì sẽ cho ra một đầu vào hợp lệ.

- Dạng tổng quát của dãy số biết A là một số nguyên dương:

$A \quad A+1 \quad A+3 \quad A+6 \quad A+10 \quad A+15$

Lưu ý: pha 2 cũng chỉ xét 6 số đầu tiên của dãy cho nên dù có nhập dãy 6 số đầu hợp lệ cộng thêm các số sau bằng dấu cách thì vẫn cho ra một đầu vào thỏa pha 2

Trường hợp input khác (khi $A = 2$):

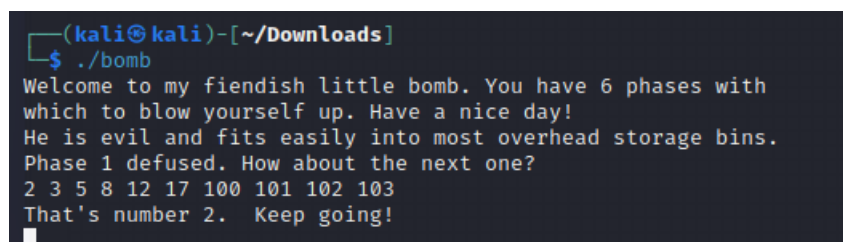


```

kali@kali: ~/Downloads
File Actions Edit View Help
(kali@kali)-[~/Downloads]
$ ./bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
He is evil and fits easily into most overhead storage bins.
Phase 1 defused. How about the next one?
2 3 5 8 12 17
That's number 2. Keep going!

```

Trường hợp input khi dãy số có 6 số đầu hợp lệ và cộng thêm các số sau tùy ý:



```

(kali@kali)-[~/Downloads]
$ ./bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
He is evil and fits easily into most overhead storage bins.
Phase 1 defused. How about the next one?
2 3 5 8 12 17 100 101 102 103
That's number 2. Keep going!

```

3. Pha 3

Xem mã giả được cung cấp:

```
int __cdecl phase_3(int a1)
{
    int result; // eax@16
    int v2; // [sp+1Ch] [bp-1Ch]@1
    int v3; // [sp+20h] [bp-18h]@1
    int v4; // [sp+24h] [bp-14h]@1
    int v5; // [sp+28h] [bp-10h]@1
    int v6; // [sp+2Ch] [bp-Ch]@1

    v6 = *MK_FP(__GS__, 20);
    v4 = 0;
    v5 = 0;
    v5 = __isoc99_sscanf(a1, "%d %d", &v2, &v3);
    if ( v5 <= 1 )
        explode_bomb();
    switch ( v2 )
    {
        case 0:
            v4 += 452;
            goto LABEL_5;
        case 1:
            LABEL_5:
                v4 -= 317;
                goto LABEL_6;
        case 2:

            LABEL_6:
                v4 += 265;
                goto LABEL_7;
        case 3:
            LABEL_7:
                v4 -= 492;
                goto LABEL_8;
        case 4:
            LABEL_8:
                v4 += 492;
                goto LABEL_9;
        case 5:
            LABEL_9:
                v4 -= 492;
                goto LABEL_10;
        case 6:
            LABEL_10:
                v4 += 492;
                break;
        case 7:
            break;
        default:
            explode_bomb();
            return result;
    }
    v4 -= 492;

    v4 -= 492;
    if ( v2 > 5 || v4 != v3 )
        explode_bomb();
    return *MK_FP(__GS__, 20) ^ v6;
}
```

Ở pha này thì nhìn vào định dạng mà hàm **scanf** gọi thì ta biết đầu vào có dạng dãy số gồm 2 số nguyên cách nhau bằng dấu cách, gọi đây là cặp số (v2 v3).

- **Manh mối đầu tiên** là v2 phải nằm trong khoảng [0, 5] để không kích hoạt bomb.

- **Manh mối thứ hai** để không kích hoạt bomb là **v3 bằng v4** => ta biết mục tiêu là tìm v4 thông qua v2 để tìm v3.

```
if ( v2 > 5 || v4 != v3 )
    explode_bomb();
return *MK_FP(__GS__, 20) ^ v6;
}
```

- Vì vậy có **6 đáp án** ứng với mỗi v2 trong khoảng trên và v3 tương ứng bằng với v4 được tính toán trong chương trình.

Dựa vào các đoạn code xử lý v4 ở mã giả trên ta có:

Code C++ nhằm tìm v4 với v2 từ 0 đến 5:

```
#include <iostream>
using namespace std;
void result(int input)
{
    int value = 0;
    int option = input;
    switch (option)
    {
        case 0:
            value += 452;
            option++;
        case 1:
            value -= 317;
            option++;
        case 2:
            value += 265;
            option++;
        case 3:
            value -= 492;
            option++;
        case 4:
            value += 492;
            option++;
        case 5:
            value -= 492;
            option++;
        default:
            value += 492;
            break;
    }
    value -= 492;
    cout << "Answer is " << input << " " << value << endl;
}

int main()
{
    result(0);
    result(1);
    result(2);
    result(3);
    result(4);
    result(5);
    return 0;
}
```

```
Microsoft Visual Studio Debug Console

Answer is 0 -92
Answer is 1 -544
Answer is 2 -227
Answer is 3 -492
Answer is 4 0
Answer is 5 -492

D:\Studying\A_Code_Space\SystemLa
To automatically close the console
le when debugging stops.
Press any key to close this window
```

Vậy có tất cả 6 cặp (v2, v3) thỏa mãn yêu cầu

Lưu ý: tương tự pha 2 thì chương trình chỉ xét 2 số đầu của đầu vào ta nhập nên đầu vào có là “0 -92” hay là “0 -92 1000” thì chương trình vẫn tiếp nhận là đúng. Vậy có 6 đầu vào chính thống và vô hạn các đầu vào không kích hoạt quả bomb.

Kiểm thử khi v2 là 0 v3 là -92:

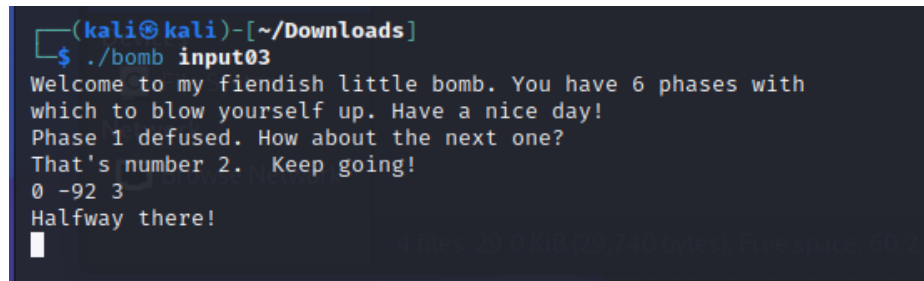


```

kali@kali: ~/Downloads
File Actions Edit View Help
(kali@kali)-[~/Downloads]
$ ./bomb input03
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Phase 1 defused. How about the next one?
That's number 2. Keep going!
0 -92
Halfway there!

```

Kiểm thử khi đầu vào là dãy số có 2 số đầu là cặp số thỏa và các số sau ngẫu nhiên:



```

(kali@kali)-[~/Downloads]
$ ./bomb input03
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Phase 1 defused. How about the next one?
That's number 2. Keep going!
0 -92 3
Halfway there!

```

4. Pha 4

Đoạn mã giả được cung cấp:

```

int __cdecl phase_4(int a1)
{
    int v2; // [sp+18h] [bp-20h]@1
    int v3; // [sp+1Ch] [bp-1Ch]@1
    int v4; // [sp+20h] [bp-18h]@1
    int v5; // [sp+24h] [bp-14h]@5
    int v6; // [sp+28h] [bp-10h]@5
    int v7; // [sp+2Ch] [bp-Ch]@1

    v7 = *MK_FP(__GS__, 20);
    v4 = __isoc99_sscanf(a1, "%d %d", &v2, &v3);
    if ( v4 != 2 || v2 < 0 || v2 > 14 )
        explode_bomb();
    v5 = 3;
    v6 = func4(v2, 0, 14);
    if ( v6 != v5 || v3 != v5 )
        explode_bomb();
    return *MK_FP(__GS__, 20) ^ v7;
}

```

Dựa vào đoạn lệnh scanf thì có thể biết đầu vào là một dãy số gồm 2 chữ số nguyên gọi cặp số này là (v2, v3)

```
if ( v4 != 2 || v2 < 0 || v2 > 14 )
    explode_bomb();
```

- Manh mới đầu tiên để không kích hoạt bomb thì số lượng input đầu vào phải là 2 và v2 nằm trong khoảng $0 \leq v2 \leq 14$.

```
v5 = 3;
v6 = func4(v2, 0, 14);
if ( v6 != v5 || v3 != v5 )
    explode_bomb();
```

- Manh mới tiếp theo là v5 = 3, v6 được lấy giá trị từ hàm func4. Cuối cùng v3 = v5 = v6 là điều kiện để không kích hoạt bomb. Ta rút ra được v3 luôn luôn bằng 3 và để v6 = 3 thì v2 ta cần tìm để là tham số cho func4 trả về kết quả là 3.

Mã giả của hàm func4:

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

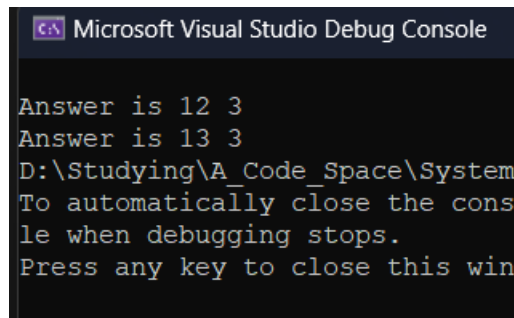
    v4 = (a3 - a2) / 2 + a2;
    if ( v4 <= a1 )
    {
        if ( v4 >= a1 )
            result = 0;
        else
            result = 2 * func4(a1, v4 + 1, a3) + 1;
    }
    else
    {
        result = 2 * func4(a1, a2, v4 - 1);
    }
    return result;
}
```

Qua đó dựng code C++ tìm giá trị v2 để v6 = 3 (= v5 = v3)

```
int func4(int a1, int a2, int a3)
{
    int result = 0;
    int 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()
{
    for (int i = 0; i <= 14; i++)
    {
        if (func4(i, 0, 14) == 3)
            cout << "\nAnswer is " << i << " " << 3;
    }

    return 0;
}
```

```

Microsoft Visual Studio Debug Console

Answer is 12 3
Answer is 13 3
D:\Studying\A_Code_Space\System
To automatically close the console
when debugging stops.
Press any key to close this window

```

Vậy ta có 2 kết quả thỏa mãn pha 4

5. Pha 5

Mã giả được cung cấp:

```

int __cdecl phase_5(int a1)
{
    int result; // eax@1
    signed int i; // [sp+4h] [bp-14h]@3
    int v3; // [sp+8h] [bp-10h]@3

    result = string_length(a1);
    if ( result != 6 )
        explode_bomb();
    v3 = 0;
    for ( i = 0; i <= 5; ++i )
    {
        result = array_2705[(int)(i + a1) & 0xF];
        v3 += result;
    }
    if ( v3 != 58 )
        explode_bomb();
    return result;
}

```

- Mạnh mối tại `result = string_length(a1)` và nếu `result` khác 6 thì sẽ kích hoạt bomb

⇒ Đầu vào là một chuỗi ký tự gồm 6 ký tự

Ta thấy vòng lặp tiếp theo sẽ cộng `result` (số nguyên) vào biến `v3`.

Sau đó so sánh `v3` và 58, nếu khác thì sẽ kích hoạt bomb.

⇒ Mấu chốt ta phải biết `result` đã được gán giá trị gì? Để sau khi cộng `v3` sẽ ra kết quả là 58.

Ta cùng xem `array_2705` chứa gì

```

.data:0804D1C0 ; int array_2705[]
.data:0804D1C0 array_2705 dd 2
.data:0804D1C4 db 0Ah
.data:0804D1C5 db 0
.data:0804D1C6 db 0
.data:0804D1C7 db 0
.data:0804D1C8 db 6
.data:0804D1C9 db 0
.data:0804D1CA db 0
.data:0804D1CB db 0
.data:0804D1CC db 1
.data:0804D1CD db 0
.data:0804D1CE db 0
.data:0804D1CF db 0
.data:0804D1D0 db 0Ch
.data:0804D1D1 db 0
.data:0804D1D2 db 0
.data:0804D1D3 db 0
.data:0804D1D4 db 10h
.data:0804D1D5 db 0

.data:0804D1D8 db 9
.data:0804D1D9 db 0
.data:0804D1DA db 0
.data:0804D1DB db 0
.data:0804D1DC db 3
.data:0804D1DD db 0
.data:0804D1DE db 0
.data:0804D1DF db 0
.data:0804D1E0 db 4
.data:0804D1E1 db 0
.data:0804D1E2 db 0
.data:0804D1E3 db 0
.data:0804D1E4 db 7
.data:0804D1E5 db 0
.data:0804D1E6 db 0
.data:0804D1E7 db 0
.data:0804D1E8 db 0Eh
.data:0804D1E9 db 0
.data:0804D1EA db 0
.data:0804D1EB db 0
.data:0804D1EC db 5
.data:0804D1ED db 0

```

```

.data:0804D1EB      db      0
.data:0804D1EC      db      5
.data:0804D1ED      db      0
.data:0804D1EE      db      0
.data:0804D1EF      db      0
.data:0804D1F0      db     0Bh
.data:0804D1F1      db      0
.data:0804D1F2      db      0
.data:0804D1F3      db      0
.data:0804D1F4      db      8
.data:0804D1F5      db      0
.data:0804D1F6      db      0
.data:0804D1F7      db      0
.data:0804D1F8      db     0Fh
.data:0804D1F9      db      0
.data:0804D1FA      db      0
.data:0804D1FB      db      0
.data:0804D1FC      db     0Dh
.data:0804D1FD      db      0
.data:0804D1FE      db      0
.data:0804D1FF      db      0
.data:0804D200      public host_table

```

Ở các vị trí $x = 0804D1C0, x+4, x+8, x+16, \dots$ chứa các số như 2, 10, 6, 1, 12, 16, ...

Vậy ta đã biết phần tử của mảng này như sau:

```
int array_2705[] = { 2, 10, 6, 1, 12, 16, 9, 3, 4, 7, 14, 5, 11, 8, 15, 13 };
```

Từ mã giả trên dựng code C++ để tìm giá trị đầu vào nhằm để v3 bằng 58 (ở đây nhóm mình chỉ vét cạn các số từ 000000 đến 999999 chưa bao gồm chữ cái):

```

#include <iostream>
using namespace std;
bool check_valid(string input)
{
    int array_2705[] = { 2, 10, 6, 1, 12, 16, 9, 3, 4, 7, 14, 5, 11, 8, 15, 13 };
    int value = 0;

    for (int i = 0; i <= 5; ++i)
    {
        int result = array_2705[(int)input[i] & 15];
        value += result;
    }

    if (value == 58)
    {
        cout << "Answer is " << input << endl;
        return 1;
    }
    return 0;
}

int main()
{
    string temp = "000000";
    while (temp != "1000000")
    {
        int i = 5;
        while (i >= 0 && temp[i] == '9')
        {
            temp[i] = '0';
            i--;
            check_valid(temp);
        }
        if (i < 0)
            break;
        temp[i]++;
    }
    return 0;
}

```

Sau khi chạy chương trình, **có rất nhiều kết quả** được tạo ra (không thể liệt kê). Kiểm thử ngẫu nhiên **10 kết quả** để vượt qua bomb thành công => Kết quả chính xác.

Sự **vô hạn** của kết quả còn đến bởi việc các số trên còn có thể **thay thế bằng chữ cái** vì chương trình xét 4 bit cuối của ký tự.

VD: Với đầu vào thỏa mãn là “**518150**” thì có thể nhận biết nhiều đầu vào thỏa mãn khác. Ta có ký tự ‘a’ có mã ascii ở dạng binary là 01100001 nên chuỗi “**5a8a50**” là một đầu vào thỏa mãn pha 5

144520	145050	145240	145420	145500	145500	145660	146560	146650	150450	150540	151110
151580	151850	152440	154050	154240	154240	154500	154660	155040	155040	155180	155400
155400	155810	155990	156460	156640	158150	158510	159590	159950	164560	164650	165460
165640	166450	166540	181550	181550	185150	195590	195950	199550	205550	214450	214540
215440	224550	225450	225540	241450	241540	242550	244150	244510	245140	245250	245410
245520	250550	251440	252450	252540	254140	254250	254410	254520	255050	255240	255420
255500	255500	255660	256560	256650	265560	265650	265650	355590	355950	359550	395550
401550	405150	405510	410550	411440	412450	412540	414140	414250	414410	414520	415050
415240	415420	415500	415500	415660	416650	421450	421540	422550	424150	424510	424510
425140	425250	425410	425520	441140	441250	441410	441520	442150	442510	444110	444580
444050	445120	445210	445480	445690	445840	445960	446590	446950	448450	448540	449560
449650	450150	450510	451050	451240	451420	451500	451660	452140	452250	452410	452410
452520	454120	454210	454480	454690	454840	454960	455010	455100	455100	455220	455670
455760	456160	456490	456570	456610	456750	456940	457560	457650	458440	459460	459640
461560	461650	464590	464950	465160	465490	465570	465610	465750	465940	466150	466510
467550	469450	469540	475560	475650	476550	484450	484540	485440	494560	494650	495460
495640	496450	496540	501450	501540	502550	504150	504510	505140	505250	505410	505520
510450	510540	511110	511580	511850	512440	514050	514240	514420	514500	514500	514660
515040	515180	515400	515400	515810	515990	516460	516640	518150	518510	519590	519950
520550	521440	522450	522540	524140	524250	524410	524520	525050	525240	525420	525500
525500	525660	526560	526650	535590	535950	540150	540510	541050	541240	541420	541420
541500	541500	541660	542140	542250	542410	542520	544120	544210	544480	544690	544840
544960	545010	545100	545100	545220	545670	545760	546160	546490	546570	546610	546750
546940	547560	547650	548440	549460	549640	550140	550250	550410	550520	551040	551180
551400	551400	551810	551990	552050	552240	552420	552500	552500	552660	553390	553950
554010	554100	554100	554220	554670	554760	555020	555200	555390	555880	555930	555930
556260	556470	556620	556740	557460	557640	558110	558580	558850	559190	559350	559530
559910	561460	561640	562560	562650	564160	564490	564570	564610	564750	564940	565260
565470	565620	565740	566140	566250	566410	566520	567450	567540	569440	574560	574650
575460	575640	576450	576540	581150	581510	584440	585110	585580	585850	588550	591590
591950	593550	594460	594640	595190	595350	595530	595910	596440	599150	599510	614560
614650	615460	615640	616450	616540	625560	625650	626550	641560	641650	644590	644950
645160	645490	645570	645610	645750	645940	646150	646510	647550	649450	649540	651460
651640	652560	652650	654160	654490	654570	654610	654750	654940	655260	655470	655620
655740	656140	656250	656410	656520	657450	657540	659440	661450	661540	662550	664150
664510	665140	665250	665410	665520	674550	675450	675540	694450	694540	695440	745560
745650	746550	754560	754650	755460	755640	756450	756540	764550	764550	765540	811550
815150	815510	844450	844540	845440	851150	851510	854440	855110	855580	855850	858550
885550	915590	915950	919550	935550	944560	944650	945460	945640	946450	946540	951590

Kiểm thử một trường hợp và sự vô hạn của đầu vào thỏa mãn (không thể liệt kê tất cả các đáp án thỏa mãn):

518150 và một biến thể khi $1 = a \Rightarrow 5a8a50$

```
kali@kali: ~/Downloads
File Actions Edit View Help
(kali@kali)-[~/Downloads]
$ ./bomb input03
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Phase 1 defused. How about the next one?
That's number 2. Keep going!
Halfway there!
So you got that one. Try this one.
518150
Good work! On to the next...
```

```
(kali@kali)-[~/Downloads]
$ ./bomb input03
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Phase 1 defused. How about the next one?
That's number 2. Keep going!
Halfway there!
So you got that one. Try this one.
5a8a50
Good work! On to the next...
```

6. Pha 6

Mã giả được cung cấp rất dài, phức tạp, nên phải chia nhỏ từ đầu để phân tích:

```

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();
    }
}

```

Chương trình gọi hàm `read_six_numbers()` như ở `phase_2` và lưu vào biến `a1` => Input đầu vào là 6 số cách nhau bằng dấu cách

Ở vòng `for`, chương trình thực hiện các điều kiện input:

- Giới hạn giá trị input nằm trong khoảng từ [0, 6]
 - Không có biến nào có giá trị giống nhau trong một lần nhập
- ⇒ Nếu không phù hợp các điều kiện trên thì bomb sẽ được kích hoạt

Tiếp theo đến phần thân xử lý các biến:

```

for ( k = 0; k <= 5; ++k )
{
    v2 = (int)&node1;
    for ( l = 1; v12[k] > 1; ++l )
        v2 = *(_DWORD*)(v2 + 8);
    v13[k] = v2;
}

```

“**v12**” là một mảng chứa **6 giá trị nguyên** được đọc từ đầu vào. Điều quan trọng cần lưu ý ở đây là giá trị của mỗi phần tử trong mảng “**v12**” chỉ nằm trong **khoảng từ 1 đến 6**. Vì vậy, trong vòng lặp đầu tiên, mỗi giá trị trong mảng “**v12**” được xử lý để tìm ra nút liên kết tương ứng trong danh sách liên kết.

Ta biết được điều đó vì vòng lặp trong này sử dụng hai biến “**v2**” và “**v13[k]**”. Biến “**v2**” được khởi tạo là địa chỉ của nút liên kết đầu tiên (“**&node1**”). Sau đó, trong vòng lặp thứ hai, giá trị của “**v2**” được cập nhật để trỏ đến nút liên kết thứ “**v12[k]**”. Nói cách khác, mỗi phần tử trong mảng “**v12**” tương ứng với **một chỉ số (index)** trong danh sách liên kết. Sau

khi vòng lặp này kết thúc, giá trị của “v13” sẽ chứa **địa chỉ của các nút** liên kết đã được xây dựng theo thứ tự tương ứng với các giá trị trong mảng ban đầu

```
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;
```

Trước khi sắp xếp, biến “v13” chứa địa chỉ của các nút liên kết đã được xây dựng. Nút đầu tiên trong danh sách (“v13[0]”) được gán cho biến “v11” và “v3”. Sau đó, trong vòng lặp, “v13[m]” được đặt vào trong trường “next” của nút hiện tại (“v3”) bằng cách sử dụng một con trỏ. Sau đó, con trỏ “v3” được di chuyển đến nút liên kết tiếp theo trong danh sách.

Ở cuối vòng lặp, sau khi tất cả các nút liên kết trong danh sách đã được nối với nhau theo **đúng thứ tự**, con trỏ của nút cuối cùng được đặt vào giá trị “NULL”.

```
v4 = v11;
for ( n = 0; n <= 4; ++n )
{
    if ( *(_DWORD *) v4 > **(_DWORD **) (v4 + 8) )
        explode_bomb();
    v4 = *(_DWORD *) (v4 + 8);
}
```

Đoạn mã giả này được sử dụng để kiểm tra xem giá trị của các nút liên kết trong danh sách có được sắp xếp tăng dần hay không. Trong đoạn mã giả này, vòng lặp duyệt qua các nút liên kết trong danh sách (“v11” là địa chỉ của nút đầu tiên), và so sánh giá trị của nút hiện tại với giá trị trong trường “next” của nút đó.

Nếu giá trị hiện tại của nút liên kết lớn hơn giá trị của nút tiếp theo (nút có địa chỉ lưu trữ trong trường “**next**”), sẽ xảy ra lỗi và chương trình sẽ kích hoạt bomb (gọi hàm “**explode_bomb()**”).

Vậy ta cần biết các node trong danh sách liên kết ra sao chứa giá trị như thế nào:

Đây là địa chỉ của các node được cung cấp:

```
data:0804D100      public node1
data:0804D0F4      public node2
.data:0804D0E8      public node3
data:0804D0DC      public node4
.data:0804D0D0      public node5
.data:0804D0C4      public node6
```

Sử dụng "examine mode" của gdb nhằm xem các địa chỉ ô nhớ trên giữ thông tin gì

```
(gdb) x/3x 0x804d100
0x804d100 <node1>: 0x000001c9 0x00000001 0x0804d0f4
(gdb) x/3x 0x804d0f4
0x804d0f4 <node2>: 0x000000d4 0x00000002 0x0804d0e8
(gdb) x/3x 0x804d0e8
0x804d0e8 <node3>: 0x000001b3 0x00000003 0x0804d0dc
(gdb) x/3x 0x804d0dc
0x804d0dc <node4>: 0x000000b5 0x00000004 0x0804d0d0
(gdb) x/3x 0x804d0d0
0x804d0d0 <node5>: 0x00000215 0x00000005 0x0804d0c4
(gdb) x/3x 0x804d0c4
0x804d0c4 <node6>: 0x000001f5 0x00000006 0x00000000
(gdb)
```

Vậy ta có bảng sau:

node 1	0x1c9
node 2	0x0d4
node 3	0x1b3
node 4	0x0b5
node5	0x215
node 6	0x1f5

Vậy với điều kiện trong vòng lặp để không kích hoạt bomb thì node đang xét nhỏ hơn node tiếp theo. Thì ta có các giá trị trong danh sách cần là:

0b5 -> 0d4 -> 1b3 -> 1c9 -> 1f5 -> 215

Hay các index tương ứng là:

4 -> 2 -> 3 -> 1 -> 6 -> 5.

Vậy đầu vào thỏa mãn cho phase 6 là: 4 2 3 1 6 5

Kiểm chứng



```
kali@kali: ~/Downloads
File Actions Edit View Help
(kali@kali)-[~/Downloads]
$ ./bomb input
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Phase 1 defused. How about the next one?
That's number 2. Keep going!
Halfway there!
So you got that one. Try this one.
Good work! On to the next...
4 2 3 1 6 5
Congratulations! You've defused the bomb!
```

7. Pha bí mật (Bonus)

YÊU CẦU CHUNG

- + Đoạn mã/Hàm xử lý tương ứng?
- + Yêu cầu về đầu vào: là chuỗi ký tự/1 số/Dãy nhiều số/Tổ hợp số và ký tự?
- + Trình bày chi tiết các bước phân tích, thuật toán/phép so sánh để kiểm tra đầu vào?
- + Kết luận về input? (Liệt kê tất cả các trường hợp input có thể có)
- + Hình ảnh minh chứng thực thi file với input tìm được.

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