BÁO CÁO THỰC HÀNH

Môn học: Lập trình hệ thống (NT209)

Lab 4 – Kỹ thuật dịch ngược nâng cao

GVHD: Đỗ Thị Hương Lan

1. THÔNG TIN CHUNG:

(Liệt kê tất cả các thành viên trong nhóm)

Lớp: NT209.N21.ATCL

STT	Họ và tên	MSSV	Email
1	Trần Thị Mỹ Huyền	21520269	21520269@gm.uit.edu.vn
2	Lâm Hải Đăng	21520682	21520682@gm.uit.edu.vn
3	Phan Thị Hồng Nhung	21521250	21521250@gm.uit.edu.vn

2. NỘI DUNG THỰC HIỆN:[1]

STT	Công việc	Kết quả tự đánh giá
1	Yêu cầu 2.1	100%
2	Yêu cầu 2.2	100%
3	Yêu cầu 2.3	100%
4	Yêu cầu 2.4	100%
5	Yêu cầu 2.5	100%
6	Yêu cầu 2.6	100%
7	Yêu cầu 2.7 (Secret)	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

1. Yêu cầu 2.1 – Pha 1

```
initialize_bomb();
puts("Welcome to my fiendish little bomb. You have 6 phases with");
puts("which to blow yourself up. Have a nice day!");
line = read_line();
phase_1(line);
phase_defused();
puts("Phase 1 defused. How about the next one?");
v4 = read_line();
phase_2(v4);
phase_defused();
puts("That's number 2.  Keep going!");
v5 = read_line();
phase_3(v5);
phase_defused();
puts("Halfway there!");
v6 = read_line();
phase_4(v6);
phase_defused();
puts("So you got that one.  Try this one.");
v7 = read_line();
phase_5(v7);
phase_defused();
puts("Good work!  On to the next...");
input = (char *)read_line();
phase_6((int)input);
phase_defused();
```

- Kiểm tra qua hàm main, ta thấy bài này sẽ có 6 phase chính được hiện rõ trong file.
- Trước hết ta sẽ kiểm tra phase_1

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

  result = strings_not_equal(a1, "For NASA, space is still a high priority.");
  if ( result )
    explode_bomb();
  return result;
}
```

- Kiểm tra mã giả cuả phase 1, ta thấy nếu kết quả của hàm result =1, chương trình sẽ chạy hàm explode_bomb(), tức không phải là đáp án đúng để gỡ bom, ngược lại sẽ trả về giá trị của result và thoát hàm chạy tiếp đến hàm phase_defused() cho phase 1.
- Hãy kiểm tra hàm strings_not_equal

```
int __cdecl strings_not_equal(_BYTE *a1, _BYTE *a2)
{
  int v2; // ebx
  _BYTE *v4; // [esp+8h] [ebp-Ch]
  _BYTE *v5; // [esp+Ch] [ebp-8h]

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

- Trước hết, ta thấy hàm này nhận 2 tham số đầu vào, với tham số a1 ở đây là một con trỏ nhằm trỏ tới địa chỉ chính xác của input đầu vào, ở đây là a2, là địa chỉ của chuỗi "For NASA, space is still a high priority.".
- Ta thấy hàm strings_not_equal trước hết so sánh độ dài của input đầu vào với chuỗi tại địa chỉ a2. Nếu bằng nhau sẽ tiếp tục thực hiện hàm, ngược lại trả về 1. Hàm tiếp tục lấy ra giá trị của input đầu vào và chuỗi ở địa chỉ a2 so sánh từng kí tự với nhau, nếu giống nhau thì tiếp tục so sánh và sẽ trả về 0 nếu 2 chuỗi hoàn toàn giống nhau. Ngược lại trả về 1 nếu có sự khác nhau
- Ở trên ta đã nói nếu giá trị trả về của hàm strings_not_equal = 1 sẽ dẫn tới việc bom nổ. Ta cần cho giá trị trả về của hàm này = 0. Tức input đầu vào của ta với chuỗi "For NASA, space is still a high priority." phải giống nhau.
- Vậy ta phát hiện được pass để defuse cho phase 1.
- **Test:**

```
┌──(kali㉿kali)-[~/Desktop/Lab04]
└─$ ./bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
For NASA, space is still a high priority.
Phase 1 defused. How about the next one?
```

2. Yêu cầu 2.2– Pha 2

```
int __cdecl phase_2(int a1)
{
  signed int i; // [sp+10h] [bp-28h]@4
  int v3; // [sp+14h] [bp-24h]@1
  int v4; // [sp+18h] [bp-20h]@2
  int v5; // [sp+2Ch] [bp-Ch]@1

  v5 = *MK_FP(__GS__, 20);
  read_six_numbers(a1, &v3);
  if ( v3 || v4 != 1 )
    explode_bomb();
  for ( i = 2; i <= 5; ++i )
  {
    if ( *(&v3 + i) != *(&v3 + i - 2) + *(&v3 + i - 1) )
      explode_bomb();
  }
  return *MK_FP(__GS__, 20) ^ v5;
}
```

- Kiểm tra mã giả phase 2, nếu v3 khác 0 hoặc v4 khác 1, thì explode_bomb() được gọi, nghĩa là chương trình sẽ bị kết thúc với một thông điệp lỗi.
- Chốt lại v3 và v4 sẽ bằng 1.
- Tiếp theo, một vòng lặp từ i = 2 đến i = 5 được thực hiện. Trong mỗi vòng lặp, điều kiện *(&v3 + i) != *(&v3 + i - 2) + *(&v3 + i - 1) được kiểm tra.
- Nếu điều kiện này không đúng, tức là số nguyên tại vị trí i không phải là tổng của hai số nguyên trước đó, thì explode_bomb() được gọi.

  Ta có:

  v3[0] = 0

  v3[1] = 1

  v3[2] = 0 + 1 = 1

  v3[3] = 1 + 1 = 2

  v3[4] = 1 + 2 = 3

  v3[5] = 2 + 3 = 5

  Dãy cần tìm: 0 1 1 2 3 5

```
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;
}
```

- Kiểm tra hàm read_six_number, các giá trị số nguyên được lưu trữ tại các địa chỉ liên tiếp trong mảng được chỉ định bởi a2, bắt đầu từ a2 và tăng dần lên 4 bytes cho mỗi số nguyên.

– Nếu result (số lượng giá trị đã đọc thành công) nhỏ hơn hoặc bằng 5, tức là không đọc được đúng sáu số nguyên, thì explode_bomb() được gọi để kết thúc chương trình với một thông điệp lỗi.

Test:

```
┌──(kali㉿kali)-[~/Downloads]
└─$ ./bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
For NASA, space is still a high priority.
Phase 1 defused. How about the next one?
0 1 1 2 3 5
That's number 2.  Keep going!
```

3. Yêu cầu 2.3– Pha 3

```
v6 = __isoc99_sscanf(a1, "%d %c %d", &v4, &v2, &v5);
if ( v6 <= 2 )
  explode_bomb();
```

– Xét mã giả phase 3, đầu vào này có định dạng "%d %c %d", với "%d" đại diện cho một số nguyên và "%c" đại diện cho một ký tự. Giá trị đọc được được lưu vào v4, v2, v5. Kết quả trả về của hàm sscanf được gán cho v6. Nếu ta nhập thiếu thì bom sẽ nổ.

```
switch ( v4 )
{
  case 0:
    v3 = 112;
    if ( v5 != 341 )
      explode_bomb();
    return result;
  case 1:
    v3 = 122;
    if ( v5 != 327 )
      explode_bomb();
    return result;
  case 2:
    v3 = 98;
    if ( v5 != 937 )
      explode_bomb();
    return result;
  case 3:
    v3 = 106;
    if ( v5 != 328 )
      explode_bomb();
    return result;
  case 4:
    v3 = 104;
    if ( v5 != 447 )
      explode_bomb();
```

```
    case 5:
      v3 = 98;
      if ( v5 != 611 )
        explode_bomb();
      return result;
    case 6:
      v3 = 115;
      if ( v5 != 293 )
        explode_bomb();
      return result;
    case 7:
      v3 = 107;
      if ( v5 != 358 )
        explode_bomb();
      return result;
    default:
      v3 = 100;
      explode_bomb();
      return result;
  }
```

– Mỗi trường hợp trong switch sẽ kiểm tra giá trị của v5 và nếu không thoả mãn điều kiện, chương trình sẽ gọi explode_bomb(). Ngoài ra dưới phần default thì bom sẽ nổ luôn. Do đó v4 là các giá trị từ 0 đến 7, v5 sẽ gán bằng với các giá trị trong từng case.

```
  if ( v3 != v2 )
    explode_bomb();
  return *MK_FP(__GS__, 20) ^ v7;
}
```

– Để bom không nổ, ta phải gán giá trị v3 bằng v2. Mặt khác v2 là giá trị %c, đổi theo bảng ASCII, ta được các trường hợp sau:

*switch v4 = 0: 0 p 341

*switch v4 = 1: 1 z 327

*switch v4 = 2: 2 b 937

*switch v4 = 3: 3 j 328

*switch v4 = 4: 4 h 447

*switch v4 = 5: 5 b 661

*switch v4 = 6: 6 s 293

*switch v4 = 7: 7 k 358

Test: Nhập 8 trường hợp đều đúng.

```
That's number 2.  Keep going!
0 p 341
Halfway there!
```

4. Yêu cầu 2.4– Pha 4

```
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", &v3, &v2)
12  if ( v4 != 2 || v2 <= 1 || v2 > 4 )
13    explode_bomb();
14  v5 = 9;
15  v6 = func4(9, v2);
16  if ( v6 != v3 )
17    explode_bomb();
18  return *MK_FP(__GS__, 20) ^ v7;
19 }
```

- Xét mã giả phase 4, giá trị đọc có đầu vào số nguyên được được lưu vào v3 và v2. Kết quả trả về của hàm sscanf được gán cho v4.
- Ở hàm if đầu tiên, để bom không nổ thì phải thỏa mãn tất cả trường hợp là 1<v2<=4, v4 =2. Nên v2 có 3 giá trị là 2, 3, 4.
- Tiếp theo gán v5 = 9. Hàm if thứ hai để bom không nổ, ta cho v6 == v3.

```
int __cdecl func4(int a1, int a2)
{
  int result; // eax@2
  int v3; // ebx@5

  if ( a1 > 0 )
  {
    if ( a1 == 1 )
    {
      result = a2;
    }
    else
    {
      v3 = func4(a1 - 1, a2) + a2;
      result = v3 + func4(a1 - 2, a2);
    }
  }
  else
  {
    result = 0;
  }
  return result;
}
```

- Xét hàm func4(9,v2), đây là hàm đệ quy nhận hai đối số a1 và a2. Kiểm tra nếu a1 lớn hơn 0:
- Nếu a1 bằng 1, tức là chúng ta đã đạt tới điều kiện dừng của đệ quy, kết quả của hàm sẽ là giá trị của a2.
- Ngược lại, chúng ta sẽ tiến hành tính toán giá trị đệ quy bằng cách:
- Gọi đệ quy func4(a1 - 1, a2) và cộng với a2.
- Gọi đệ quy func4(a1 - 2, a2).
- Cộng hai giá trị đệ quy lại với nhau và lưu vào biến v3.
- Kết quả của hàm là giá trị của v3.
- Nếu a1 không lớn hơn 0, tức là a1 là một giá trị không hợp lệ, kết quả của hàm sẽ là 0.
- Tóm lại, ta sẽ được công thức v3 = 88*v2.

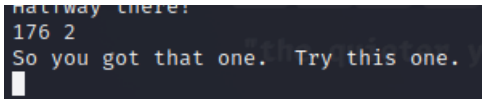
***Input:**

v2 = 2 => v3 = 176 => Nhập 176 2

v2 = 3 => v3 = 264 => Nhập 264 3

v2 = 4 => v3 = 352 => Nhập 352 4

Test: Các trường hợp đều đúng



5. Yêu cầu 2.5– Pha 5

```
unsigned int __cdecl phase_5(int a1)
{
  int v2; // [esp+14h] [ebp-24h] BYREF
  int v3; // [esp+18h] [ebp-20h] BYREF
  int v4; // [esp+1Ch] [ebp-1Ch]
  int v5; // [esp+20h] [ebp-18h]
  int v6; // [esp+24h] [ebp-14h]
  int v7; // [esp+28h] [ebp-10h]
  unsigned int v8; // [esp+2Ch] [ebp-Ch]

  v8 = __readgsdword(0x14u);
  v6 = __isoc99_sscanf(a1, "%d %d", &v2, &v3);
  if ( v6 <= 1 )
    explode_bomb();
  v2 &= 15u;
  v7 = v2;
  v4 = 0;
  v5 = 0;
  while ( v2 != 15 )
  {
    ++v4;
    v2 = array_2704[v2];
    v5 += v2;
  }
  if ( v4 != 15 || v5 != v3 )
    explode_bomb();
  return __readgsdword(0x14u) ^ v8;
}
```

- Xét mã giả phase 5, giá trị đọc có đầu vào số nguyên được được lưu vào v2 và v3. Kết quả trả về của hàm sscanf được gán cho v6.
- Ở hàm if đầu tiên, để bom không nổ thì phải thỏa mãn trường hợp v6 > 1
- Tiếp theo ta AND v2 với 15. Và gán v7 = v2, v4 =0, v5 = 0.
- Ta kiểm tra điều kiện của hàm if ở dưới để cho bomb không nổ thì v4 = 15 và v5 = v3. Ta trở lên kiểm tra vòng lặp while() để tìm v4 và v5.
- Ở vòng lặp while điều kiện để kết thúc là v2 = 15, và ta thấy v4 tăng lên 1 mỗi vòng lặp. Suy ra vòng lặp phải chạy được 15 lần rồi kết thúc. Mỗi vòng lặp thì v2 đổi giá trị thành array_2704[v2] và v5 += v2.
- Ta kiểm tra mảng array_2704 để tìm được v2 sao cho vòng lặp chạy được 15 lần thì v2 = 15.

```
.data:0804D1C0 ; int array_2704[16]
.data:0804D1C0 array_2704      dd 0Ah
.data:0804D1C4                 db    2
.data:0804D1C5                 db    0
.data:0804D1C6                 db    0
.data:0804D1C7                 db    0
.data:0804D1C8                 db  0Eh
.data:0804D1C9                 db    0
.data:0804D1CA                 db    0
.data:0804D1CB                 db    0
.data:0804D1CC                 db    7
.data:0804D1CD                 db    0
.data:0804D1CE                 db    0
.data:0804D1CF                 db    0
.data:0804D1D0                 db    8
.data:0804D1D1                 db    0
.data:0804D1D2                 db    0
.data:0804D1D3                 db    0
.data:0804D1D4                 db  0Ch
.data:0804D1D5                 db    0
.data:0804D1D6                 db    0
.data:0804D1D7                 db    0
.data:0804D1D8                 db  0Fh
.data:0804D1D9                 db    0
.data:0804D1DA                 db    0
.data:0804D1DB                 db    0
.data:0804D1DC                 db  0Bh
.data:0804D1DD                 db    0
.data:0804D1DE                 db    0
.data:0804D1DF                 db    0
.data:0804D1E0                 db    0
.data:0804D1E1                 db    0
.data:0804D1E2                 db    0
.data:0804D1E3                 db    0
.data:0804D1E4                 db    4
.data:0804D1E8                 db    1
.data:0804D1E9                 db    0
.data:0804D1EA                 db    0
.data:0804D1EB                 db    0
.data:0804D1EC                 db  0Dh
.data:0804D1ED                 db    0
.data:0804D1EE                 db    0
.data:0804D1EF                 db    0
.data:0804D1F0                 db    3
.data:0804D1F1                 db    0
.data:0804D1F2                 db    0
.data:0804D1F3                 db    0
.data:0804D1F4                 db    9
.data:0804D1F5                 db    0
.data:0804D1F6                 db    0
.data:0804D1F7                 db    0
.data:0804D1F8                 db    6
.data:0804D1F9                 db    0
.data:0804D1FA                 db    0
.data:0804D1FB                 db    0
.data:0804D1FC                 db    5
.data:0804D1FD                 db    0
.data:0804D1FE                 db    0
.data:0804D1FF                 db    0
```

– Mảng array_2704 này là mảng kiểu số nguyên (int) chứa 16 phần tử. Vì kiểu int 4 bytes nên ta có thể biết được vị trí các phần tử trong mảng.

array_2704[0] = 0x0A (10 decimal)

array_2704[1] = 0x02 (2 decimal)

array_2704[2] = 0x0E (14 decimal)

array_2704[3] = 0x07 (7 decimal)

array_2704[4] = 0x08 (8 decimal)

array_2704[5] = 0x0C (12 decimal)

array_2704[6] = 0x0F (15 decimal)

array_2704[7] = 0x0B (11 decimal)

array_2704[8] = 0x00 (0 decimal)

array_2704[9] = 0x04 (4 decimal)

array_2704[10] = 0x01 (1 decimal)

array_2704[11] = 0x0D (13 decimal)

array_2704[12] = 0x03 (3 decimal)

array_2704[13] = 0x09 (9 decimal)

array_2704[14] = 0x06 (6 decimal)

array_2704[15] = 0x05 (5 decimal)

– Ta bắt đầu từ v2 = 15 chạy ngược lại 15 lần để tìm được v4 = 15 và tìm v2 ban đầu.

v2 = 15 -> v2 = 6 -> v2 = 14 -> v2 = 2

-> v2 = 1 -> v2 = 10 -> v2 = 0 -> v2 = 8

-> v2 = 4 -> v2= 9 -> v2 = 13 -> v2 = 11

-> v2 = 7 -> v2 = 3 -> v2 = 12 -> v2 = 5

- Biết được v2 ban đầu là 5. Ta tính giá trị v5 sau khi chạy hết vòng lặp là 115. Suy ra được giá trị v3 = 115.
- Ta cùng có thể nhập giá trị v2 khác miễn thỏa tính chất v2 & 0xF = 5

***Input: 5 115**

Test:

```
5 115
Good work!  On to the next ...
```

6. **Yêu cầu 2.6 - Pha 6**

```
1  unsigned int __cdecl phase_6(int a1)
2  {
3    _DWORD *v2; // [esp+1Ch] [ebp-4Ch]
4    int v3; // [esp+1Ch] [ebp-4Ch]
5    int v4; // [esp+1Ch] [ebp-4Ch]
6    int i; // [esp+20h] [ebp-48h]
7    int k; // [esp+20h] [ebp-48h]
8    int n; // [esp+20h] [ebp-48h]
9    int ii; // [esp+20h] [ebp-48h]
10   int j; // [esp+24h] [ebp-44h]
11   int m; // [esp+24h] [ebp-44h]
12   int v11; // [esp+28h] [ebp-40h]
13   int v12[6]; // [esp+2Ch] [ebp-3Ch] BYREF
14   int v13[6]; // [esp+44h] [ebp-24h]
15   unsigned int v14; // [esp+5Ch] [ebp-Ch]
16
17   v14 = __readgsdword(0x14u);
18   read_six_numbers(a1, (int)v12);
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   }
```

Xét mã giả phase 6, giá trị đọc có đầu vào là 6 số nguyên được lưu vào mảng v12. Xét vòng for đầu tiên

- Ở hàm if đầu tiên để bom không nổ thì các giá trị trong mảng v12 phải nằm trong {1,2,3,4,5,6}. Vòng for
- Vòng for (dòng 23) kết hợp với hàm if để cho các phần tử trong mảng v12 không được giống nhau thì bom mới không nổ

Suy ra ta phải nhập vào 6 số khác nhau thuộc tập {1,2,3,4,5,6}

```
29   for ( k = 0; k <= 5; ++k )
30   {
31     v2 = &node1;
32     for ( m = 1; v12[k] > m; ++m )
33       v2 = (_DWORD *)v2[2];
34     v13[k] = (int)v2;
35   }
36   v11 = v13[0];
37   v3 = v13[0];
```

Xét vòng for thứ hai

Xét các node:

- Ta có thể thấy có 6 node trong file bomb và mỗi node bao gồm 12 bytes. Với 4 bytes đầu là giá trị "data" của node, 4 bytes tiếp theo là số thứ tự "i" của node, và 4 bytes cuối là địa chỉ chỉ đến node tiếp theo. => Đây chính là linked list với struct được định nghĩa như sau:

```
struct node
{
        int data;
        int i;
        struct node *next;
};
```

- v2 là con trỏ kiểu _DWORD, và được gán địa chỉ của node1.
- Vòng for (dòng 32) chạy từ m = 1, đến khi v12[k] <= m. Ta thấy mỗi lần lặp, v2 sẽ được gán bằng phần tử thứ 2 của mảng v2. Với v2 là địa chỉ của node 1, giá trị được gán vào mỗi vòng lặp chính là địa chỉ trỏ tới node tiếp theo của linked list đã được định nghĩa trước đó (4 bytes cuối của mỗi node trong linked list – biến node *next)

```
.data:0804D0E8                 public node3
.data:0804D0E8 node3           db  5Eh ; ^
.data:0804D0E9                 db    0
.data:0804D0EA                 db    0
.data:0804D0EB                 db    0
.data:0804D0EC                 db    3
.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
.data:0804D0F4                 public node2
.data:0804D0F4 node2           db 0F6h
.data:0804D0F5                 db    0
.data:0804D0F6                 db    0
.data:0804D0F7                 db    0
.data:0804D0F8                 db    2
.data:0804D0F9                 db    0
.data:0804D0FA                 db    0
.data:0804D0FB                 db    0
.data:0804D0FC                 db 0E8h
.data:0804D0FD                 db 0D0h
.data:0804D0FE                 db    4
.data:0804D0FF                 db    8
.data:0804D100                 public node1
.data:0804D100 node1           db 0F6h                 ; DATA XREF: phase_6+17↑o
.data:0804D101                 db    0
.data:0804D102                 db    0
.data:0804D103                 db    0
.data:0804D104                 db    1
.data:0804D105                 db    0
.data:0804D106                 db    0
.data:0804D107                 db    0
.data:0804D108                 db 0F4h
.data:0804D109                 db 0D0h
.data:0804D10A                 db    4
.data:0804D10B                 db    8
```

- Sau đó gán giá trị của v2 vào mảng v13.
- Ta cứ chạy vòng lặp for thứ hai này cho đến khi quét được hết các phần tử trong mảng v12.
- Ngắn gọn vòng lặp trên nhằm từng phần tử thứ i trong mảng v13 với v13[i] = linked list [v12[i]] với v12 là mảng input đầu vào.

Ta gán v11 = v13[0] và v3 = v13[0] với v13[0] tùy thuộc vào chuỗi input

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

Xét vòng for thứ ba

- Ta gán giá trị của phần tử thứ n của mảng v13 vào địa chỉ (v3+8)
- Vòng lặp này nhằm duyệt qua linked list vừa mới được tạo ở vòng lặp trên (mảng v13)

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

- Ở vòng lặp này, để bom không nổ thì giá trị của node trước phải lớn hơn hoặc bằng node sau, ta có thể hiểu như sau: node -> data >= (node -> next) -> data. Với giá trị của v4 bằng phần tử đầu tiên của mảng v13 được tạo ở trên.

```
        hex       dec
node1: 0xf6   => 246
node2: 0xf6   => 246
node3: 0x5e   => 94
node4: 0x194  => 404
node5: 0x2bc  => 700
node6: 0x278  => 632
```

- Sau khi dò các giá trị của các node thì ta thu được bảng giá trị hex trên. Để thỏa mãn điều kiện của hàm if trên thì ta sẽ sắp xếp các node theo thứ tự sau: node5 > node 6 > node4 > node 1 (node 2) >= node2 (node1) > node3 cũng như theo thứ tự input để thỏa mãn đề bài.

***Input**: 5 6 4 1 2 3 hoặc 5 6 4 2 1 3

Test:

```
5 6 4 1 2 3
Congratulations! You've defused the bomb!
```

```
5 6 4 2 1 3
Congratulations! You've defused the bomb!
```

7. Yêu cầu 2.7 Phase Secret

```
unsigned int phase_defused()
{
  char v1; // [esp+0h] [ebp-68h] BYREF
  char v2; // [esp+4h] [ebp-64h] BYREF
  int v3; // [esp+8h] [ebp-60h]
  char v4[80]; // [esp+Ch] [ebp-5Ch] BYREF
  unsigned int v5; // [esp+5Ch] [ebp-Ch]

  v5 = __readgsdword(0x14u);
  if ( num_input_strings == 6 )
  {
    v3 = __isoc99_sscanf(&unk_804D530, "%d %d %s", &v1, &v2, v4);
    if ( v3 == 3 && !strings_not_equal(v4, "DrEvil") )
    {
      puts("Curses, you've found the secret phase!");
      puts("But finding it and solving it are quite different...");
      secret_phase();
    }
    puts("Congratulations! You've defused the bomb!");
  }
  return __readgsdword(0x14u) ^ v5;
}
```

- Ở bên dưới phase 4, ta có thấy có hàm phase_defused(), bên trong hàm này thực hiện việc check input của phase 4 1 lần nữa với input là 2 số nguyên với 1 chuỗi.
- Hàm kiểm tra nếu input của phase 4 đủ 3 giá trị nêu trên và chuỗi bằng với giá trị "DrEvil" sẽ mở được secret_phase.
- Vậy ta chỉ cần sau phase 4, ta nhập thêm chuỗi "DrEvil" sẽ mở được secret_phase sau khi nhập hết được 6 phase.
- **Test:**

```
┌──(kali㉿kali)-[~/Desktop/Lab04]
└─$ ./bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
For NASA, space is still a high priority.
Phase 1 defused. How about the next one?
0 1 1 2 3 5
That's number 2.  Keep going!
0 p 341
Halfway there!
176 2 DrEvil
So you got that one.  Try this one.
5 115
Good work!  On to the next...
5 6 4 1 2 3
Curses, you've found the secret phase!
But finding it and solving it are quite different...
```

- Ta xét tiếp hàm secret_phase

```
unsigned int secret_phase()
{
  char *nptr; // [esp+4h] [ebp-14h]
  int v2; // [esp+8h] [ebp-10h]

  nptr = (char *)read_line();
  v2 = atoi(nptr);
  if ( v2 <= 0 || v2 > 1001 )
    explode_bomb();
  if ( fun7(&n1, v2) != 4 )
    explode_bomb();
  puts("Wow! You've defused the secret stage!");
  return phase_defused();
}
```

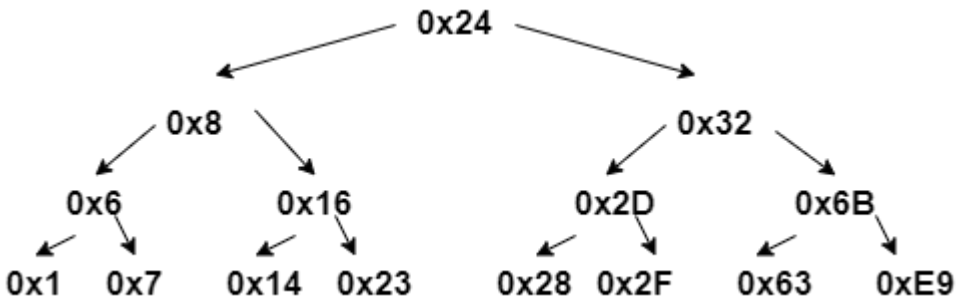
- Ta có thể thấy hàm secret_phase nhận 1 input là 1 con trỏ là chuỗi.
- Hàm atoi nhận input nptr để chuyển từ định dạng string sang int.
- Tiếp theo xét input người dùng được chuyển sang số phải nằm trong khoảng từ 1 – 1000
- Tiếp theo chương trình gọi đến hàm fun7 nhận 2 input gồm input đầu vào người dùng nhập đổi thành int và địa chỉ biến n1.
- Bây giờ ta sẽ kiểm tra biến n1

```
0804D1B4 n1             db  24h
0804D1B5                db    0
0804D1B6                db    0
0804D1B7                db    0
0804D1B8                db 0A8h
0804D1B9                db 0D1h
0804D1BA                db    4
0804D1BB                db    8
0804D1BC                db  9Ch
0804D1BD                db 0D1h
0804D1BE                db    4
0804D1BF                db    8
```

- n1 gồm 12 bytes. Với 4 bytes đầu là một giá trị. 4 bytes sau chỉ đến một địa chỉ, 4 bytes cuối chỉ đến một địa chỉ nữa tuần tự là n21 và n22

```
:0804D19C n22           db  32h ; 2
:0804D19D               db    0
:0804D19E               db    0
:0804D19F               db    0
:0804D1A0               db  84h
:0804D1A1               db 0D1h
:0804D1A2               db    4
:0804D1A3               db    8
:0804D1A4               db  6Ch ; l
:0804D1A5               db 0D1h
:0804D1A6               db    4
:0804D1A7               db    8
:0804D1A8               public n21
:0804D1A8 n21           db    8
:0804D1A9               db    0
:0804D1AA               db    0
:0804D1AB               db    0
:0804D1AC               db  78h ; x
:0804D1AD               db 0D1h
:0804D1AE               db    4
:0804D1AF               db    8
:0804D1B0               db  90h
:0804D1B1               db 0D1h
:0804D1B2               db    4
:0804D1B3               db    8
```

- Tương tự với n1, n21 và n22 cũng có kiểu struct tương tự. Vậy ta có thể hình dung ra được đây là một binary tree như sau:



- Xét hàm fun7:

```
int __cdecl fun7(_DWORD *a1, int a2)
{
  if ( !a1 )
    return -1;
  if ( *a1 > a2 )
    return 2 * fun7(a1[1], a2);
  if ( *a1 == a2 )
    return 0;
  return 2 * fun7(a1[2], a2) + 1;
}
```

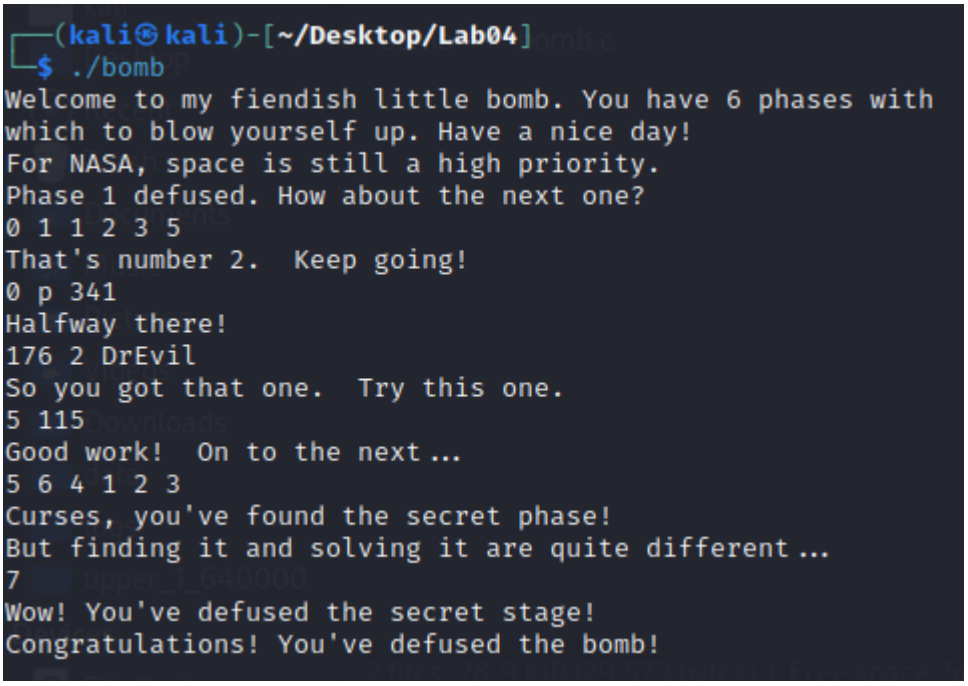
- Hàm fun 7 nhận 2 input gồm input đầu vào người dùng nhập đổi thành int và địa chỉ biến n1. Nếu kết quả trả về của hàm fun 7 !=4 bomb sẽ nổ. Tức ta phải nhập input sao cho kết quả trả về của hàm bằng 4.
- Hàm fun 7 là một hàm đệ quy có thể ghi rõ lại theo cấu trúc binary tree như sau:

```
int fun7 (node *n, int x)
{
        if (n==null)
                return -1;
        if (n->data < = x)
                {
                if (n->data == value)
                        return 0;
                return 2* fun7(n->right, x) + 1;
                }
        else
                return 2* fun7 (n->left, x);
}
```

- Trong các node trong binary tree, ta cần chọn 1 node sao cho thứ tự return của hàm fun7 có thể như lần lượt như bên dưới (từ dưới ngược lên) nhằm đảm bảo kết quả trả về sẽ bằng 4. Tức ta phải chọn làm sao cho thứ tự duyệt cây từ nút root -> nút trái -> nút trái -> nút phải.

```
return fun7 (node *n, int x)
        return 2* fun7 (node *n, int x)
                return 2* fun7 (node *n , int x)
                        return 2* fun7 (node *n, int x) +1
                                return 0;
```

- Xét binary tree, ta thấy node mang giá trị 0x7 thỏa điều kiện trả về trên. Vậy đáp án cho phase secret này sẽ là số 7 (Lưu ý input nhập vào phải là hex được đổi thành decimal vì program đổi từ chuỗi sang int)
- **Kiểm tra kết quả:**


(kali㉿kali)-[~/Desktop/Lab04]
$ ./bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
For NASA, space is still a high priority.
Phase 1 defused. How about the next one?
0 1 1 2 3 5
That's number 2. Keep going!
0 p 341
Halfway there!
176 2 DrEvil
So you got that one. Try this one.
5 115
Good work! On to the next...
5 6 4 1 2 3
Curses, you've found the secret phase!
But finding it and solving it are quite different...
7
Wow! You've defused the secret stage!
Congratulations! You've defused the bomb!

HẾT