Lab 5

BÁO CÁO BÀI THỰC HÀNH SỐ 5

REVERSE ENGINEERING

(cont)

Môn học: Lập trình Hệ thống (NT209)

Sinh viên thực hiện	22521303 – Nguyễn Đức Tấn 22521541 – Thái Ngọc Diễm Trinh 22521501 – Phan Nguyễn Nhật Trâm
Thời gian thực hiện	08/05/2024

Mã giả của hàm main được IDAPro phân tích như sau:

```
  puts("\n[*] Phase 1\n- Hint: Numbers are always magical!");
  v3 = read_line();
  phase1(v3);
  defuse_bomb();
  puts("\n[*] Phase 2\n- Hint: You must answer your secret question!");
  v4 = read_line();
  phase2(v4);
  defuse_bomb();
  puts("\n[*] Phase 3\n- Hint: Many cases make everything so confusing.");
  v5 = read_line();
  phase3(v5);
  defuse_bomb();
  puts("\n[*] Phase 4\n- Hint: Let's dig in to recursive function :)");
  v6 = read_line();
  phase4(v6);
  defuse_bomb();
  puts("\n[*] Phase 5\n-Hint: No hint is also a hint :)");
  s = (char *)read_line();
  phase5(s);
  defuse_bomb();
  return 0;
}
```

Trước mỗi phase đều lấy dữ liệu từ bàn phím rồi truyền vào phase đó.

PHASE 1

Mã assembly:

```
.text:0804898B                push    ebp
.text:0804898C                mov     ebp, esp
.text:0804898E                sub     esp, 38h
.text:08048991                lea     eax, [ebp+var_2C]
.text:08048994                add     eax, 14h
.text:08048997                push    eax
.text:08048998                lea     eax, [ebp+var_2C]
.text:0804899B                add     eax, 10h
.text:0804899E                push    eax
.text:0804899F                lea     eax, [ebp+var_2C]
.text:080489A2                add     eax, 0Ch
.text:080489A5                push    eax
.text:080489A6                lea     eax, [ebp+var_2C]
.text:080489A9                add     eax, 8
.text:080489AC                push    eax
.text:080489AD                lea     eax, [ebp+var_2C]
.text:080489B0                add     eax, 4
.text:080489B3                push    eax
.text:080489B4                lea     eax, [ebp+var_2C]
.text:080489B7                push    eax
.text:080489B8                push    offset aDDDDDD  ; "%d %d %d %d %d %d"
.text:080489BD                push    [ebp+arg_0]
.text:080489C0                call    ___isoc99_sscanf
.text:080489C5                add     esp, 20h
.text:080489C8                mov     [ebp+var_10], eax
.text:080489CB                cmp     [ebp+var_10], 6
.text:080489CF                jz      short loc_80489D6
.text:080489D1                call    explode_bomb
.text:080489D6 ; ---------------------------------------------------------------------------
.text:080489D6
.text:080489D6 loc_80489D6:                           ; CODE XREF: phase1+44j
.text:080489D6                mov     [ebp+var_14], 1
.text:080489DD                mov     eax, [ebp+var_2C]
.text:080489E0                cmp     eax, [ebp+var_14]
.text:080489E3                jge     short loc_80489EA
.text:080489E5                call    explode_bomb
.text:080489EA ; ---------------------------------------------------------------------------
.text:080489EA
.text:080489EA loc_80489EA:                           ; CODE XREF: phase1+58j
.text:080489EA                mov     [ebp+var_C], 1
.text:080489F1                jmp     short loc_8048A16
.text:080489F3 ; ---------------------------------------------------------------------------
.text:080489F3
.text:080489F3 loc_80489F3:                           ; CODE XREF: phase1+8Fj
.text:080489F3                mov     eax, [ebp+var_C]
.text:080489F6                mov     eax, [ebp+eax*4+var_2C]
.text:080489FA                mov     edx, [ebp+var_C]
```

```
.text:080489FD                sub     edx, 1
.text:08048A00                mov     ecx, [ebp+edx*4+var_2C]
.text:08048A04                mov     edx, [ebp+var_C]
.text:08048A07                add     edx, ecx
.text:08048A09                cmp     eax, edx
.text:08048A0B                jz      short loc_8048A12
.text:08048A0D                call    explode_bomb
.text:08048A12 ; ---------------------------------------------------------------------------
.text:08048A12
.text:08048A12 loc_8048A12:                           ; CODE XREF: phase1+80j
.text:08048A12                add     [ebp+var_C], 1
.text:08048A16
.text:08048A16 loc_8048A16:                           ; CODE XREF: phase1+66j
.text:08048A16                cmp     [ebp+var_C], 5
.text:08048A1A                jle     short loc_80489F3
.text:08048A1C                nop
.text:08048A1D                leave
.text:08048A1E                retn
.text:08048A1E phase1         endp
.text:08048A1E
.text:08048A1F
```

Phân tích mã giả của phase 1:

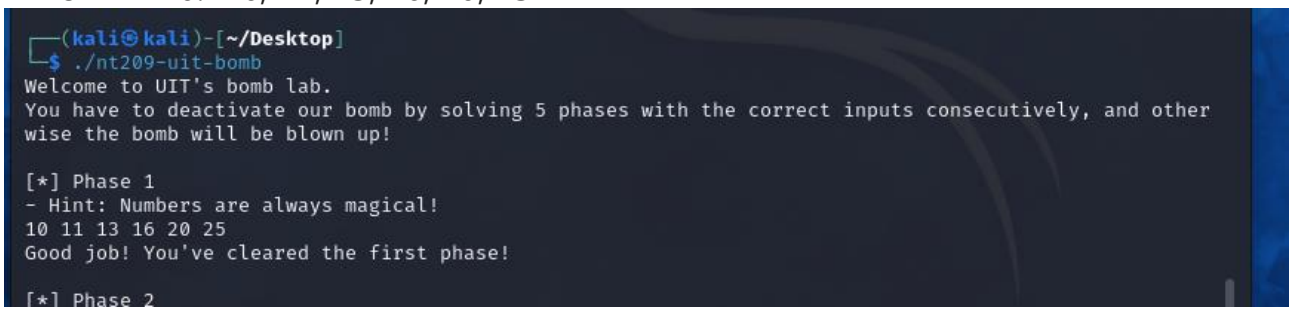
```
int __cdecl phase1(int a1)
{
  int result; // eax@3
  int v2[6]; // [sp+Ch] [bp-2Ch]@1
  int v3; // [sp+24h] [bp-14h]@3
  int v4; // [sp+28h] [bp-10h]@1
  int i; // [sp+2Ch] [bp-Ch]@5

  v4 = __isoc99_sscanf(a1, "%d %d %d %d %d %d", v2, &v2[1], &v2[2], &v2[3],
&v2[4], &v2[5]);
  if ( v4 != 6 )
    explode_bomb();
  v3 = 1;
  result = v2[0];
  if ( v2[0] < 1 )
    explode_bomb();
  for ( i = 1; i <= 5; ++i )
  {
    result = v2[i];
    if ( result != v2[i - 1] + i )
      explode_bomb();
  }
  return result;
}
```

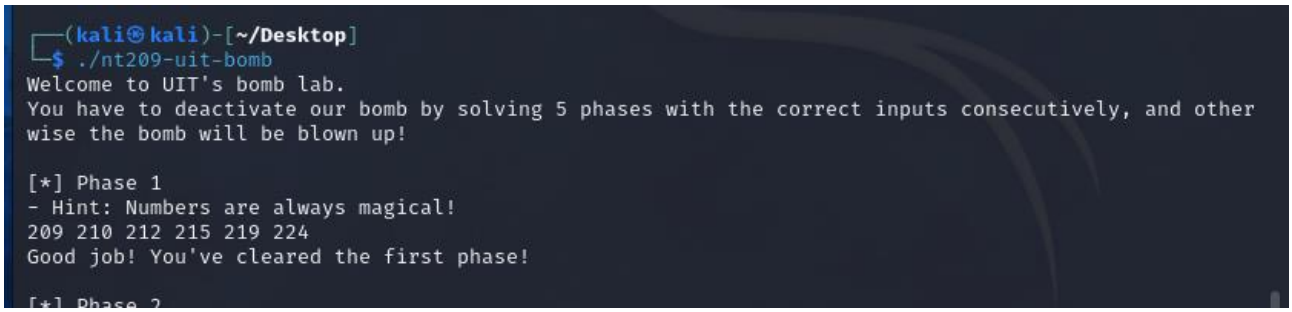
Giải thích cách tìm input:

- 6 số nguyên
- Điều kiện ràng buộc:
  - Số nguyên đầu tiên nhập vào phải lớn hơn 0.
  - v2[i] = v2[i-1] + i
- Input của pha có thể là:
  - Với v2[0] = 1 --> 1 2 4 7 11 16
  - Với v2[0] = 2 --> 2 3 5 8 12 17
  - Với số nguyên n bất kỳ lớn hơn 0: dãy số cần tìm là: n, n + 1, n + 3, n + 6, n + 10, n + 15
- Kết quả thực thi pha:

+ Với n = 10: 10, 11, 13, 16, 20, 25



+ Với n = 209: 209 210 212 215 219 224



PHASE 2:

Mã assembly:

```
.text:08048A1F                push    ebp
.text:08048A20                mov     ebp, esp
.text:08048A22                sub     esp, 28h
.text:08048A25                mov     [ebp+var_C], 0Dh
.text:08048A2C                mov     eax, [ebp+var_C]
.text:08048A2F                mov     eax, QUESTIONS[eax*4]
.text:08048A36                mov     [ebp+var_10], eax
.text:08048A39                mov     eax, [ebp+var_C]
.text:08048A3C                mov     eax, QA_MAP[eax*4]
.text:08048A43                mov     [ebp+var_14], eax
.text:08048A46                mov     eax, [ebp+var_14]
.text:08048A49                mov     eax, ANSWERS[eax*4]
.text:08048A50                mov     [ebp+s2], eax
.text:08048A53                push    [ebp+arg_0]
.text:08048A56                call    transfer
.text:08048A5B                add     esp, 4
.text:08048A5E                mov     [ebp+s1], eax
.text:08048A61                mov     eax, [ebp+s2]
.text:08048A64                movzx   eax, byte ptr [eax]
.text:08048A67                test    al, al
.text:08048A69                jz      short loc_8048A80
.text:08048A6B                sub     esp, 8
.text:08048A6E                push    [ebp+s2]        ; s2
.text:08048A71                push    [ebp+s1]        ; s1
.text:08048A74                call    is_equal
.text:08048A79                add     esp, 10h
.text:08048A7C                test    eax, eax
.text:08048A7E                jnz     short loc_8048A85
.text:08048A80
.text:08048A80 loc_8048A80:                           ; CODE XREF:phase2+4Aj
.text:08048A80                call    explode_bomb
.text:08048A85 ; ---------------------------------------------------------------
.text:08048A85
.text:08048A85 loc_8048A85:                           ; CODE XREF:phase2+5Fj
.text:08048A85                nop
.text:08048A86                leave
.text:08048A87                retn
.text:08048A87 phase2         endp
```

Mã giả:

```
int __cdecl phase2(int a1)
{
  char *v1; // ST28_4@1
  int result; // eax@2
  char *s1; // [sp+Ch] [bp-1Ch]@1
  char *s2; // [sp+10h] [bp-18h]@1
```

```
  v1 = QUESTIONS[13];
  s2 = ANSWERS[*(&QA_MAP + 13)];
  s1 = (char *)transfer(a1);
  if ( !*s2 || (result = is_equal(s1, s2)) == 0 )
    explode_bomb();
  return result;
}
```

Hàm phase 2 nhận tham số a1 từ bàn phím người dùng

Hàm explode_bomb() sẽ nổ khi kết quả của phép so sánh 2 chuỗi s1, s2 khác 0. Có nghĩa là chuỗi s1 (được nhập vào) phải là kết quả của câu hỏi v1 (QUESTION[13]).

Dựa vào phân tích mã bằng IDA Pro, ta tìm được QUESTION[13] = *Which city has 3/7 of a chicken and 2/3 of a cat and 1/2 of a goat? (Capitalize the first letter).*

```
.data:0804B05D                  db    0
.data:0804B05E                  db    0
.data:0804B05F                  db    0
.data:0804B060                  public QUESTIONS
.data:0804B060 QUESTIONS        dd offset aWhatIsTheClass ; DATA XREF: phase2+10↑r
.data:0804B060                                            ; "What is the class code of this course?"
.data:0804B064                  dd offset aWhatIsTheFullE ; "What is the full English name of our un"...
.data:0804B068                  dd offset aCompleteTheDom ; "Complete the domain of our faculty: htt"...
.data:0804B06C                  dd offset aWhatIsYourMajo ; "What is your major in English? (Capital"...
.data:0804B070                  dd offset aWhatIsTheEmail ; "What is the email address of your pract"...
.data:0804B074                  dd offset aWhatIsThePhone ; "What is the phone number of our univers"...
.data:0804B078                  dd offset aWhatIsTheMainL ; "What is the main language used in this "...
.data:0804B07C                  dd offset aWhatIsTheWeekd ; "What is the weekday that this class tak"...
.data:0804B080                  dd offset aEnterTheCurren ; "Enter the current date using the format"...
.data:0804B084                  dd offset aIAmAnOddNumber ; "I am an odd number. Take away one lette"...
.data:0804B088                  dd offset aWhatIsTheNameO ; "What is the name of the analyzed execut"...
.data:0804B08C                  dd offset aWhatIsYourNati ; "What is your nationality?"
.data:0804B090                  dd offset aWhatIsTheEstab ; "What is the establishment date of UIT ("...
.data:0804B094                  dd offset aWhichCityHas37 ; "Which city has 3/7 of a chicken and 2/3"...
.data:0804B098                  dd offset aWhatIsTheVietn ; "What is the Vietnamese name (without ac"...
.data:0804B09C                  dd offset aMyVehicleRegis ; "My vehicle registration plate starts wi"...
.data:0804B0A0                  dd offset aWhatIsTheLarge ; "What is the largest country in the worl"...
.data:0804B0A4                  dd offset aWhatWordIsSpel ; "What word is spelled incorrectly in eve"...
.data:0804B0A8                  dd offset aIAmOnTheWall_T ; "I am on the wall. Thanks to me, you can"...
.data:0804B0AC                  dd offset aWhatIsTheLonge ; "What is the longest wall in the world? "...
.data:0804B0B0                  align 20h
.data:0804B0C0                  public QA_MAP
```

Đáp án của câu hỏi là Chicago

Kết quả thực thi phase 2:

```
[*] Phase 2
- Hint: You must answer your secret question!
Chicago
Two phases have been solved. Keep going!

[*] Phase 3
```

PHASE 3:

Mã assembly nhận được:

```
.text:08048A88                 push    ebp
.text:08048A89                 mov     ebp, esp
.text:08048A8B                 sub     esp, 18h
.text:08048A8E                 mov     [ebp+var_C], 0
.text:08048A95                 mov     [ebp+var_10], 0
.text:08048A9C                 lea     eax, [ebp+var_18]
.text:08048A9F                 push    eax
.text:08048AA0                 lea     eax, [ebp+var_14]
.text:08048AA3                 push    eax
.text:08048AA4                 push    offset aDD      ; "%d %d"
.text:08048AA9                 push    [ebp+arg_0]
.text:08048AAC                 call    ___isoc99_sscanf
.text:08048AB1                 add     esp, 10h
.text:08048AB4                 mov     [ebp+var_10], eax
.text:08048AB7                 cmp     [ebp+var_10], 1
.text:08048ABB                 jg      short loc_8048AC2
.text:08048ABD                 call    explode_bomb
.text:08048AC2 ; ---------------------------------------------------------------
.text:08048AC2
.text:08048AC2 loc_8048AC2:                            ; CODE XREF: phase3+33j
.text:08048AC2                 mov     eax, [ebp+var_14]
.text:08048AC5                 cmp     eax, 7          ; switch 8 cases
.text:08048AC8                 ja      short loc_8048B0D ; jumptable 08048AD1
default case
.text:08048ACA                 mov     eax, ds:off_804976C[eax*4]
.text:08048AD1                 jmp     eax             ; switch jump
.text:08048AD3 ; ---------------------------------------------------------------
.text:08048AD3
.text:08048AD3 loc_8048AD3:                            ; CODE XREF: phase3+49j
.text:08048AD3                                         ; DATA XREF:
.rodata:off_804976Co
.text:08048AD3                 add     [ebp+var_C], 1B8h ; jumptable 08048AD1
case 0
.text:08048ADA
.text:08048ADA loc_8048ADA:                            ; CODE XREF: phase3+49j
.text:08048ADA                                         ; DATA XREF:
.rodata:off_804976Co
.text:08048ADA                 sub     [ebp+var_C], 22Bh ; jumptable 08048AD1
case 1
.text:08048AE1
.text:08048AE1 loc_8048AE1:                            ; CODE XREF: phase3+49j
.text:08048AE1                                         ; DATA XREF:
.rodata:off_804976Co
.text:08048AE1                 add     [ebp+var_C], 28Bh ; jumptable 08048AD1
case 2
.text:08048AE8
```

```
.text:08048AE8 loc_8048AE8:                            ; CODE XREF: phase3+49j
.text:08048AE8                                         ; DATA XREF:
.rodata:off_804976Co
.text:08048AE8                 sub     [ebp+var_C], 1F5h ; jumptable 08048AD1
case 3
.text:08048AEF
.text:08048AEF loc_8048AEF:                            ; CODE XREF: phase3+49j
.text:08048AEF                                         ; DATA XREF:
.rodata:off_804976Co
.text:08048AEF                 add     [ebp+var_C], 1F5h ; jumptable 08048AD1
case 4
.text:08048AF6
.text:08048AF6 loc_8048AF6:                            ; CODE XREF: phase3+49j
.text:08048AF6                                         ; DATA XREF:
.rodata:off_804976Co
.text:08048AF6                 sub     [ebp+var_C], 1F5h ; jumptable 08048AD1
case 5
.text:08048AFD
.text:08048AFD loc_8048AFD:                            ; CODE XREF: phase3+49j
.text:08048AFD                                         ; DATA XREF:
.rodata:off_804976Co
.text:08048AFD                 add     [ebp+var_C], 1F5h ; jumptable 08048AD1
case 6
.text:08048B04
.text:08048B04 loc_8048B04:                            ; CODE XREF: phase3+49j
.text:08048B04                                         ; DATA XREF:
.rodata:off_804976Co
.text:08048B04                 sub     [ebp+var_C], 1F5h ; jumptable 08048AD1
case 7
.text:08048B0B                 jmp     short loc_8048B12
.text:08048B0D ; ---------------------------------------------------------------
.text:08048B0D
.text:08048B0D loc_8048B0D:                            ; CODE XREF: phase3+40j
.text:08048B0D                 call    explode_bomb    ; jumptable 08048AD1
default case
.text:08048B12 ; ---------------------------------------------------------------
.text:08048B12
.text:08048B12 loc_8048B12:                            ; CODE XREF: phase3+83j
.text:08048B12                 mov     eax, [ebp+var_14]
.text:08048B15                 cmp     eax, 5
.text:08048B18                 jg      short loc_8048B22
.text:08048B1A                 mov     eax, [ebp+var_18]
.text:08048B1D                 cmp     [ebp+var_C], eax
.text:08048B20                 jz      short loc_8048B27
.text:08048B22
.text:08048B22 loc_8048B22:                            ; CODE XREF: phase3+90j
.text:08048B22                 call    explode_bomb
.text:08048B27 ; ---------------------------------------------------------------
.text:08048B27
```

```
.text:08048B27 loc_8048B27:                            ; CODE XREF: phase3+98j
.text:08048B27                 nop
.text:08048B28                 leave
.text:08048B29                 retn
.text:08048B29 phase3          endp
```

Mã giả:

```
int __cdecl phase3(int a1)
{
  int result; // eax@14
  int v2; // [sp+0h] [bp-18h]@1
  int v3; // [sp+4h] [bp-14h]@1
  int v4; // [sp+8h] [bp-10h]@1
  int v5; // [sp+Ch] [bp-Ch]@1

  v5 = 0;
  v4 = 0;
  v4 = __isoc99_sscanf(a1, "%d %d", &v3, &v2);
  if ( v4 <= 1 )
    explode_bomb();
  switch ( v3 )
  {
    case 0:
      v5 += 440;
      goto LABEL_5;
    case 1:
LABEL_5:
      v5 -= 555;
      goto LABEL_6;
    case 2:
LABEL_6:
      v5 += 651;
      goto LABEL_7;
    case 3:
LABEL_7:
      v5 -= 501;
      goto LABEL_8;
    case 4:
LABEL_8:
      v5 += 501;
      goto LABEL_9;
    case 5: // v3 = 5 // ng dung nhap
LABEL_9:
      v5 -= 501; // v5 = -501
      goto LABEL_10;
    case 6:
LABEL_10:
      v5 += 501; // v5 = 0
```

```
      break; // thoat ra
    case 7:
      break;
    default:
      explode_bomb();
      return result;
  }
  v5 -= 501; // v5 = -501
  if ( v3 > 5 || (result = v2, v5 != v2) ) // v2 == v5
    explode_bomb();
  return result;
}
```

Giải thích cách tìm input:

1. Định dạng của input: 2 số nguyên (v3 và v2)
2. Điều kiện ràng buộc:
    a. Số nguyên đầu tiên nhập vào phải nhỏ hơn hoặc bằng 5
    b. Số nguyên thứ hai bắt buộc bằng v5
3. Input của phase 3 có thể là 5 -501

- Các input có thể khác là: **(4,0); (3,-501), (2,150), (1, -405), (0, 35)**

4. Kết quả thực thi:

```
[*] Phase 1
- Hint: Numbers are always magical!
1 2 4 7 11 16
Good job! You've cleared the first phase!

[*] Phase 2
- Hint: You must answer your secret question!
Chicago
Two phases have been solved. Keep going!

[*] Phase 3
- Hint: Many cases make everything so confusing.
5 -501
You've beaten another phase, that's great. What about the fourth one?

[*] Phase 4
- Hint: Let's dig in to recursive function :)
```

```
Two phases have been solved. Keep going!

[*] Phase 3
- Hint: Many cases make everything so confusing.
4 0
You've beaten another phase, that's great. What about the fourth one?

[*] Phase 4
- Hint: Let's dig in to recursive function :)
```

```
Chicago
Two phases have been solved. Keep going!

[*] Phase 3
- Hint: Many cases make everything so confusing.
3 -501
You've beaten another phase, that's great. What about the fourth one?

[*] Phase 4
- Hint: Let's dig in to recursive function :)
```

```
[*] Phase 3
- Hint: Many cases make everything so confusing.
2 150
You've beaten another phase, that's great. What about the fourth one?
```

```
[*] Phase 3
- Hint: Many cases make everything so confusing.
1 -405
You've beaten another phase, that's great. What about the fourth one?
```

```
Two phases have been solved. Keep going!

[*] Phase 3
- Hint: Many cases make everything so confusing.
0 35
You've beaten another phase, that's great. What about the fourth one?
```

PHASE 4:

Mã assembly nhận được:

```
.text:08048B82                 push    ebp
.text:08048B83                 mov     ebp, esp
.text:08048B85                 sub     esp, 28h
.text:08048B88                 lea     eax, [ebp+var_18]
.text:08048B8B                 push    eax
.text:08048B8C                 lea     eax, [ebp+var_1C]
.text:08048B8F                 push    eax
.text:08048B90                 push    offset aDD      ; "%d %d"
.text:08048B95                 push    [ebp+arg_0]
.text:08048B98                 call    ___isoc99_sscanf
.text:08048B9D                 add     esp, 10h
.text:08048BA0                 mov     [ebp+var_C], eax
.text:08048BA3                 cmp     [ebp+var_C], 2
.text:08048BA7                 jnz     short loc_8048BB9
.text:08048BA9                 mov     eax, [ebp+var_18]
.text:08048BAC                 cmp     eax, 1
.text:08048BAF                 jle     short loc_8048BB9
.text:08048BB1                 mov     eax, [ebp+var_18]
.text:08048BB4                 cmp     eax, 4
.text:08048BB7                 jle     short loc_8048BBE
.text:08048BB9
.text:08048BB9 loc_8048BB9:                            ; CODE XREF: phase4+25j
.text:08048BB9                                         ; phase4+2Dj
.text:08048BB9                 call    explode_bomb
.text:08048BBE ; ---------------------------------------------------------------------------
.text:08048BBE
.text:08048BBE loc_8048BBE:                            ; CODE XREF: phase4+35j
.text:08048BBE                 mov     [ebp+var_10], 7
.text:08048BC5                 mov     eax, [ebp+var_18]
.text:08048BC8                 sub     esp, 8
.text:08048BCB                 push    eax
.text:08048BCC                 push    [ebp+var_10]
.text:08048BCF                 call    func4
.text:08048BD4                 add     esp, 10h
.text:08048BD7                 mov     [ebp+var_14], eax
.text:08048BDA                 mov     eax, [ebp+var_1C]
.text:08048BDD                 cmp     [ebp+var_14], eax
.text:08048BE0                 jz      short loc_8048BE7
.text:08048BE2                 call    explode_bomb
.text:08048BE7 ; ---------------------------------------------------------------------------
.text:08048BE7
.text:08048BE7 loc_8048BE7:                            ; CODE XREF: phase4+5Ej
.text:08048BE7                 nop
```

```
.text:08048BE8                 leave
.text:08048BE9                 retn
.text:08048BE9 phase4          endp
```

Mã giả:

```
int __cdecl phase4(int a1)
{
  int result; // eax@5
  int v2; // [sp+Ch] [bp-1Ch]@1
  int v3; // [sp+10h] [bp-18h]@1
  int v4; // [sp+14h] [bp-14h]@5
  int v5; // [sp+18h] [bp-10h]@5
  int v6; // [sp+1Ch] [bp-Ch]@1

  v6 = __isoc99_sscanf(a1, "%d %d", &v2, &v3);
  if ( v6 != 2 || v3 <= 1 || v3 > 4 )
    explode_bomb();
  v5 = 7;
  v4 = func4(7, v3);
  result = v2;
  if ( v4 != v2 )
    explode_bomb();
  return result;
}
```

1. Định dạng input: 2 số nguyên (v2 và v3)
2. Điều kiện ràng buộc:
- Số nguyên thứ 2 (v3): lớn hơn 1 và nhỏ hơn bằng 4 (1 < v3 <= 4)
- Số nguyên đầu tiên phải là kết quả của hàm func4(7, v3)

Phân tích hàm func4 bằng mã giả:

```
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;
}
```

Minh hoạ hàm func4 bằng code C và tiến hành chạy với các input v3 lần lượt là 2, 3, 4:

Ta thu được kết quả:

```
C:\Users\tieul\Document\1_HK4\NT209_LTHT\Thuc_hanh\Lab5>a.exe
2: func4(7, 2) = 66
3: func4(7, 3) = 99
4: func4(7, 4) = 132
```

3. Input của phase 4 lần lượt là: (66, 2); (99, 3); (132, 4)

4. Minh chứng:

```
[*] Phase 4
- Hint: Let's dig in to recursive function :)
66 2
Awesome! Only one phase left!
```

```
[*] Phase 4
- Hint: Let's dig in to recursive function :)
99 3
Awesome! Only one phase left!
```

```
[*] Phase 4
- Hint: Let's dig in to recursive function :)
132 4
Awesome! Only one phase left!

[*] Phase 5
```

PHASE 5:

Mã assembly:

```
.text:08048BEA var_10          = dword ptr -10h
.text:08048BEA var_C           = dword ptr -0Ch
.text:08048BEA s               = dword ptr  8
.text:08048BEA
.text:08048BEA                 push    ebp
.text:08048BEB                 mov     ebp, esp
.text:08048BED                 sub     esp, 18h
.text:08048BF0                 sub     esp, 0Ch
.text:08048BF3                 push    [ebp+s]         ; s
.text:08048BF6                 call    _strlen
.text:08048BFB                 add     esp, 10h
.text:08048BFE                 mov     [ebp+var_14], eax
.text:08048C01                 cmp     [ebp+var_14], 6
.text:08048C05                 jz      short loc_8048C0C
.text:08048C07                 call    explode_bomb
.text:08048C0C ; ---------------------------------------------------------------------------
.text:08048C0C
.text:08048C0C loc_8048C0C:                            ; CODE XREF: phase5+1Bj
.text:08048C0C                 mov     [ebp+var_10], 0
.text:08048C13                 mov     [ebp+var_C], 0
.text:08048C1A                 jmp     short loc_8048C3B
.text:08048C1C ; ---------------------------------------------------------------------------
.text:08048C1C
.text:08048C1C loc_8048C1C:                            ; CODE XREF: phase5+55j
.text:08048C1C                 mov     edx, [ebp+var_C]
.text:08048C1F                 mov     eax, [ebp+s]
.text:08048C22                 add     eax, edx
.text:08048C24                 movzx   eax, byte ptr [eax]
.text:08048C27                 movsx   eax, al
.text:08048C2A                 and     eax, 0Fh
.text:08048C2D                 mov     eax, array_3852[eax*4]
.text:08048C34                 add     [ebp+var_10], eax
.text:08048C37                 add     [ebp+var_C], 1
.text:08048C3B
.text:08048C3B loc_8048C3B:                            ; CODE XREF: phase5+30j
.text:08048C3B                 cmp     [ebp+var_C], 5
.text:08048C3F                 jle     short loc_8048C1C
.text:08048C41                 cmp     [ebp+var_10], 30h
.text:08048C45                 jz      short loc_8048C4C
.text:08048C47                 call    explode_bomb
.text:08048C4C ; ---------------------------------------------------------------------------
.text:08048C4C
.text:08048C4C loc_8048C4C:                            ; CODE XREF: phase5+5Bj
.text:08048C4C                 nop
```

```
.text:08048C4D                leave
.text:08048C4E                retn
.text:08048C4E phase5         endp
```

Mã giả:

```
size_t __cdecl phase5(char *s)
{
  size_t result; // eax@1
  int v2; // [sp+8h] [bp-10h]@3
  signed int i; // [sp+Ch] [bp-Ch]@3

  result = strlen(s);
  if ( result != 6 )
    explode_bomb();
  v2 = 0;
  for ( i = 0; i <= 5; ++i )
  {
    result = array_3852[s[i] & 0xF];
    v2 += result;
  }
  if ( v2 != 48 )
    explode_bomb();
  return result;
}
```

1. Định dạng của input: Một chuỗi có độ dài là 6 (6 kí tự)
2. Điều kiện ràng buộc của input:

- Nhập vào 1 chuỗi có độ dài là 6.
- Duyệt từng kí tự trong chuỗi. Thực hiện phép AND giữa kí tự đó với 0xF (hay lấy 4 bit cuối của kí tự). Khi đó 4 bit này sẽ là chỉ số của mảng array_3852[].
- Truy xuất vị trí array_3852[i] tương ứng và lưu vào biến result. Cộng dồn biến result vào biến v2.
- Nếu v2 = 48 ⇨ thỏa yêu cầu, ngược lại bom sẽ nổ.
- Mảng array_3852[] có các phần tử sau: 2, 10, 6, 1, 12, 16, 9, 3, 4, 7, 14, 5, 11, 8

```
.data:0804B1A0 ; int array_3852[]
.data:0804B1A0 array_3852      dd 2, 0Ah, 6, 1, 0Ch, 10h, 9, 3, 4, 7, 0Eh, 5,
0Bh, 8
```

3. Kết luận về input của pha (liệt kê hoặc mô tả tất cả các input thỏa mãn).

- Viết 1 chương trình C++ để liệt kê một vài input thỏa mãn. Vì có nhiều chuỗi thỏa mãn nên sẽ lưu các chuỗi trong 1 file “input.txt”:

```
#include <iostream>
#include <fstream>
```

```
#include <string>
using namespace std;

// Hàm thực hiện logic của Phase 5
bool phase5(const string &s)
{
    if (s.length() != 6) // Kiểm tra độ dài chuỗi
        return false;

    int result = 0;
    int v2 = 0;
    int array[] = {2, 10, 6, 1, 12, 16, 9, 3, 4, 7, 14, 5, 11, 8};

    // Thực hiện vòng lặp từ 0 đến 5
    for (int i = 0; i < 6; ++i)
    {
        // Lấy 4 bit cuối của s[i]
        int index = s[i] & 0xF;
        // Gán giá trị của mảng array tại chỉ số index vào result
        result = array[index];
        // Cộng result vào v2
        v2 += result;
    }

    // Kiểm tra nếu v2 khác 48 thì bom nổ
    if (v2 != 48)
    {
        return false;
    }

    return true;
}

int main()
{
    // Mở file để ghi
    ofstream outputFile("input.txt");

    // Duyệt tất cả các số từ 100000 đến 999999
    for (int a = 100000; a <= 999999; a++)
    {
        string input = to_string(a);
        if (phase5(input))
        {
            // Ghi vào file các chuỗi thỏa mãn yêu cầu
            outputFile << input << ", ";
        }
    }
```

```
    // Đóng file
    outputFile.close();

    return 0;
}
```

- Kết quả thu được một số chuỗi sau:

```
100055, 100144, 100245, 100254, 100414, 100425, 100441, 100452,
100505, 100524, 100542, 100550, 100566, 100656, 100665, 101044,
101112, 101121, 101148, 101169, 101184, 101196, 101211, 101258,
101285, 101356, 101365, 101404, 101418, 101440, 101481, 101499,
101528, 101536, 101563, 101579, 101582, 101597, 101619, 101635,
101653, 101691, 101759, 101795, 101814, 101825, 101841, 101852,
101916, 101949, 101957, 101961, 101975, 101994, 102045, 102054,
102111, 102158, 102185, 102244, 102405, 102424, 102442, 102450,
102466, 102504, 102518, 102540, 102581, 102599, 102646, 102664,
102815, 102851, 102959, 102995, 103156, 103165, 103459, 103495,
103516, 103549, 103557, 103561, 103575, 103594, 103615, 103651,
103755, 103945, 103954, 104014, 104025, 104041, 104052, 104104,
104118, 104140, 104181, 104199, 104205, 104224, 104242, 104250,
104266, 104359, 104395, 104401, 104410, 104422, 104467, 104476,
104502, 104520, 104539, 104588, 104593, 104626, 104647, 104662,
104674, 104746, 104764, 104811, 104858, 104885, 104919, 104935,
104953, 104991, 105005, 105024, 105042, 105050, 105066, 105128,
105136, 105163, 105179, 105182, 105197, 105204, 105218, 105240,
105281, 105299, 105316, 105349, 105357, 105361, 105375, 105394,
```

4. Hình ảnh chụp kết quả thực thi với 1 hoặc nhiều input đã tìm thấy với file nt209-uit-bomb.

- Chạy file chương trình:

+ 100144:

```
[*] Phase 5
-Hint: No hint is also a hint :)
100144
Amazing bomb solvers, the bomb has been deactived. Enjoy your day :))
```

+ 999985:

```
[*] Phase 5
-Hint: No hint is also a hint :)
999985
Amazing bomb solvers, the bomb has been deactived. Enjoy your day :))
```