BÁO CÁO THỰC HÀNH

LẬP TRÌNH HỆ THỐNG

Tên bài Thực hành: Lab 05 - Lớp: NT209.P12.ANTT

Giáo viên hướng dẫn: Đỗ Thị Thu Hiền

Họ và tên sinh viên	MSSV
Nguyễn Trần Minh Khôi	23520780
Lê Đăng Khôi	23520766
Vương Thành Đạt	23520281

Bài 1:

- Hàm explode_bomb():

```
void __noreturn explode_bomb()
{
  puts("BOMB!!!!\nThe bomb has blown up. Try again.");
  exit(0);
}
```

Phase 1:

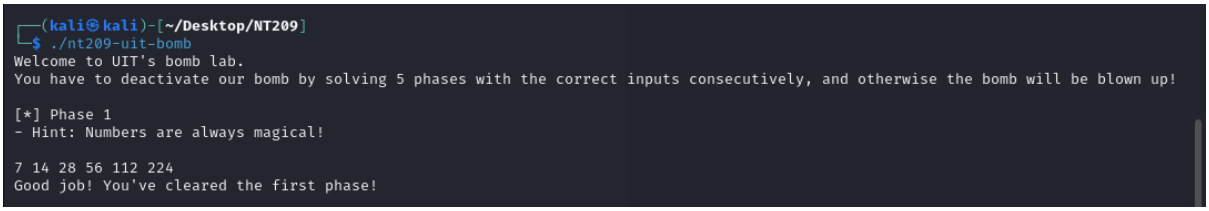
- Hình ảnh:

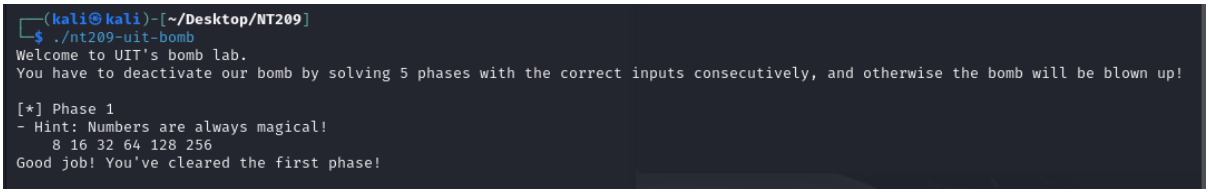
```
int __cdecl phase1(int a1)
{
  int result; // eax@3
  int v2[6]; // [sp+Ch] [bp-2Ch]@1
  int v3[6]; // [sp+10h] [bp-28h]@1
  int v4[6]; // [sp+14h] [bp-24h]@1
  int v5[6]; // [sp+18h] [bp-20h]@1
  int v6[6]; // [sp+1Ch] [bp-1Ch]@1
  int v7[6]; // [sp+20h] [bp-18h]@1
  int v8; // [sp+24h] [bp-14h]@3
  int v9; // [sp+28h] [bp-10h]@1
  int i; // [sp+2Ch] [bp-Ch]@5

  v9 = __isoc99_sscanf(a1, "%d %d %d %d %d %d", v2, v3, v4, v5, v6, v7);
  if ( v9 != 6 )
    explode_bomb();
  v8 = 7;
  result = v2[0];
  if ( v2[0] < 7 )
    explode_bomb();
  for ( i = 1; i <= 5; ++i )
  {
    result = v2[i];
    if ( result != 2 * v2[i - 1] )
      explode_bomb();
  }
  return result;
```

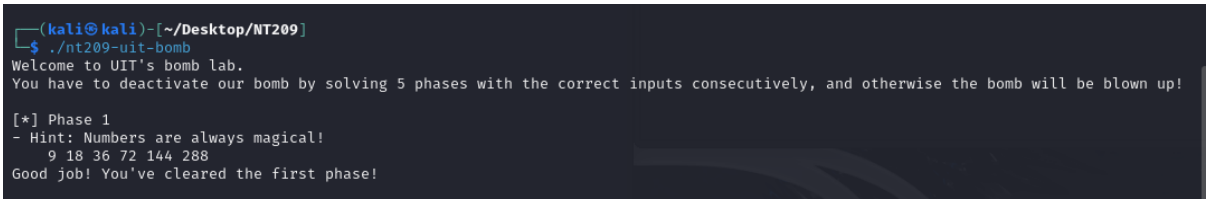
- Tiêu chí làm bài làm sao để tránh trả về hàm explode_bomb(), nên ta phải lấy trường hợp ngược lại các toán tử.
- Giải thích:
  - Khai báo biến và nhập liệu:
    - Đầu tiên, hàm sẽ thực hiện khai báo các biến và mảng cần thiết để thực hiện lưu trữ, sau đó hàm sẽ thực hiện nhập dữ liệu bằng hàm **v9 = __isoc99_sscanf(a1, "%d %d %d %d %d %d", v2, v3, v4, v5, v6, v7)** , nhận chuỗi a1 với %d là định dạng kiểu int cho từng mảng được lưu trữ như v2, v3, v4, v5, v6, v7, v9 lưu số giá trị đọc được.

- Điều kiện:
  - v9 = 6 để không dẫn đến hàm explode_bomb().
  - Gán giá trị v8 = 7.
  - Gán giá trị đầu tiên trong mảng v2, tức v2[0] = result.
  - Kiểm tra điều kiện v2[0] >= 7 để tránh hàm explode_bomb().
- Vòng lặp For:
  - Chỉ số i chạy từ 1 tới 5, trong đó biến result được nạp giá trị từ mảng chỉ số v2[1] tới v2[5].
  - Kiểm tra các giá trị của mảng v2 có chỉ số từ 1 tới 5 có bằng 2 lần biến result để tránh hàm explode_bomb().
- Trả về kết quả:
  - Sau khi thoát khỏi vòng lặp For, sẽ trả về giá trị của biến result.
- Vậy ta có thể kết luận các kết quả result như sau:
  - 7 14 28 56 112 224
  - 8 16 32 64 128 256
  - 9 18 36 72 144 288
  - x * pow(2, i) với x >= 7, i thuộc [0; 5].

- Kết quả:
  - Vậy kết quả nhỏ nhất trong các kết quả tìm được là:
    - 7 14 28 56 112 224







Bài 2:

- Hàm is_equal():

```
BOOL __cdecl is_equal(char *s1, char *s2)
{
  return strcmp(s1, s2) == 0;
}
```

- Hàm transfer():

```
int __cdecl transfer(int a1)
{
  char v2; // [sp+Ah] [bp-6h]@8
  char v3; // [sp+Bh] [bp-5h]@2
  int i; // [sp+Ch] [bp-4h]@1

  for ( i = 0; *(_BYTE *)(i + a1); ++i )
  {
    v3 = *(_BYTE *)(i + a1);
    if ( (v3 <= 96 || v3 > 122) && (v3 <= 64 || v3 > 90) )
    {
      if ( v3 > 47 && v3 <= 57 )
        v3 = (v3 - 48 + 5) % 10 + 48;
    }
    else
    {
      if ( v3 <= 96 || v3 > 122 )
        v2 = 65;
      else
        v2 = 97;
      v3 = (v3 - v2 + 5) % 26 + v2;
    }
    *(_BYTE *)(a1 + i) = v3;
  }
  return a1;
}
```

- Biến cục bộ:
  - Khai báo v2, v3 với kiểu ký tự (char), biến i với kiểu số nguyên (int).
  - Trong đó:
    - v2 lưu giá trị **ASCII** là ký tự bắt đầu trong bảng chữ cái (**A hay a**).
    - v3 lưu giá trị ký tự hiện tại của chuỗi.
    - i là biến đếm, duyệt qua từng ký tự trong chuỗi.
- Vòng lặp:
  - Duyệt qua từng phần tử ký tự trong chuỗi, kết thúc khi gặp ký tự '**\0**' (ký tự dùng để kết thúc chuỗi).
  - Mỗi ký tự được thay đổi tại chuỗi vì **v3 = *(_BYTE *)(i + a1)** sau mỗi vòng lặp.
- Kiểm tra ký tự:
- Ký tự là không chữ cái:
  - if ((v3 <= 96 || v3 > 122) && (v3 <= 64 || v3 > 90))
    - Trong ASCII: số 96 là ký tự ' ` ', số 122 ký tự là '**z**', số 64 ký tự là '**@**', số 90 ký tự là '**Z**'.
    - Tức là điều kiền đang xét v3 không nằm trong khoảng [**a** ; **z**] và khoảng [**A** ; **Z**].
  - **if (v3 > 47 && v3 <= 57)**
    - Trong ASCII: số 47 là ký tự ' / ', số 57 là ký tự ' 9 '.
    - Tức là điều kiện đang xét lồng tiếp v3 nằm trong khoảng [0 ; 9]. Điều này khẳng định v3 đang là số.
  - **v3 = (v3 - 48 + 5) % 10 + 48**

        - Thỏa mãn điều kiện, thực hiện phép dịch vòng (+ 5) trong khoảng [0 ; 9], sau đó chuyển về khoảng số.
- Ký tự là chữ cái:
    - **if (v3 <= 96 || v3 > 122)**
        - Trong ASCII: số 96 là ký tự ‘ ` ‘, số 122 ký tự là ‘ **z** ’. Tức là điều kiện đang xét v3 nằm trong khoảng [**a** ; **z**] và khoảng [**A** ; **Z**].
    - **v3 = (v3 - v2 + 5) % 26 + v2**
        - Thực hiện phép dịch vòng (+ 5) trong bảng chữ cái tương ứng.
- Cập nhật ký tự:
    - ***(_BYTE *)(a1 + i) = v3**
        - Sau khi xử lý chuỗi, ký tự được ghi vào vị trí tương ứng trong chuỗi đầu vào
- Trả về giá trị:
    - Sau khi hoàn tất, hàm trả về a1, là địa chỉ của chuỗi đã mã hóa.

Phase 2:

```
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[6];
  s2 = ANSWERS[*(&QA_MAP + 6)];
  s1 = (char *)transfer(a1);
  if ( !*s2 || (result = is_equal(s1, s2)) == 0 )
    explode_bomb();
  return result;
}
```

- Giải thích:
- Khai báo:
    - Khai báo ba con trỏ kiểu char, với mỗi phần tử khi con trỏ trỏ tới có 4 bytes
- Trích xuất mảng **QUESTIONS, ANSWERS, QA_MAP**:
    - **v1 = QUESTIONS[6]** tức là v1 sẽ truy xuất đến mảng **QUESTIONS** có chỉ số 6.
    - **s2 = ANSWERS[*(&QA_MAP + 6)]** sẽ truy xuất câu trả lời trong mảng **ANSWERS** dựa trên giá trị được ánh xạ thông qua **QA_MAP**.
- Xử lý đầu vào:
    - Hàm transfer(a1) dùng để chuyển đổi dữ liệu đầu vào của phase2 thông qua hàm transfer() và ép kiểu con trỏ kiểu char, sau đó lưu vào s1.
- Điều kiện:
    - Để tránh bomb từ hàm explode_bomb(), ta phải kiểm tra ***s2 != NULL** hoặc result là kết quả của so sánh hai chuỗi s1, s2, phải làm sao để 2 chuỗi khác nhau.
- Trả về kết quả:
    - Nếu vượt qua hết hàm explode_bomb() thì sẽ trả về giá trị result, mà result được gán bằng hàm is_equal(s1, s2).

- Theo như giải thích ở trên, v1 sẽ truy xuất tới câu hỏi thứ 7.

```
.data:0804B060 QUESTIONS       dd offset aMyVehicleRegis ; DATA XREF: phase2+10↑r
.data:0804B060                                           ; "My vehicle registration plate starts wi"...
.data:0804B064                 dd offset aWhatIsTheEngli ; "What is the English name of our course?"...
.data:0804B068                 dd offset aWhatIsTheCapit ; "What is the capital of Thailand?"
.data:0804B06C                 dd offset aWhichSeasonHas ; "Which season has cherry blossoms?"
.data:0804B070                 dd offset aIsabellaSParen ; "Isabella's parents have four children. "...
.data:0804B074                 dd offset aWhichCountryIs ; "Which country is the Lion city in South"...
.data:0804B078                 dd offset aIf2024IsTheYea ; "If 2024 is the year of Dragon, what ani"...
.data:0804B07C                 dd offset aEnterTheCurren ; "Enter the current date using the format"...
.data:0804B080                 dd offset aWhichProvinceI ; "Which province in Vietnam has the most "...
.data:0804B084                 dd offset aWhatIsTheLarge ; "What is the largest country in the worl"...
.data:0804B088                 dd offset aWhatWordIsSpel ; "What word is spelled incorrectly in eve"...
.data:0804B08C                 dd offset aWhatIsYourNati ; "What is your nationality?"
.data:0804B090                 dd offset aWhatIsThePhone ; "What is the phone number of our univers"...
```

```
.rodata:08049014 aIf2024IsTheYea db 'If 2024 is the year of Dragon, what animal will 2025 be the year '
.rodata:08049014                                         ; DATA XREF: .data:0804B078↓o
```

- s2 trong mảng ANSWERS sẽ là "**Xsfpj**"

```
.data:0804B160 ANSWERS         dd offset aGnsmIztsl     ; DATA XREF: phase2+2A↑r
.data:0804B160                                          ; "Gnsm Iztsl"
.data:0804B164                 dd offset aHtruzyjwXdxyjr ; "Htruzyjw Xdxyjr Uwtlwfrrnsl"
.data:0804B168                 dd offset aGfslptp       ; "Gfslptp"
.data:0804B16C                 dd offset aXuwnsl        ; "Xuwnsl"
.data:0804B170                 dd offset aNxfgjqqf      ; "Nxfgjqqf"
.data:0804B174                 dd offset aXnslfutwj     ; "Xnslfutwj"
.data:0804B178                 dd offset aXsfpj         ; "Xsfpj"
```

- Ta cần giải mã chuỗi này về ban đầu trước khi bị mã hóa: **Snake**
- Code **C++**:

```cpp
#include <bits/stdc++.h>
using namespace std;

void reverse(string &s)
{
    char v2, v3;
    for (size_t i = 0; i < s.length(); ++i)
    {
        v3 = s[i];
        if ((v3 >= 'A' && v3 <= 'Z')) { // Ký tự in hoa
            v2 = 'A'; // Base cho chữ in hoa
            v3 = (v3 - v2 - 5 + 26) % 26 + v2; // Dịch ngược 5
        }
        else if ((v3 >= 'a' && v3 <= 'z')) { // Ký tự in thường
            v2 = 'a'; // Base cho chữ in thường
            v3 = (v3 - v2 - 5 + 26) % 26 + v2; // Dịch ngược 5
        }
        else if (v3 >= '0' && v3 <= '9') { // Ký tự số
            v3 = (v3 - '0' - 5 + 10) % 10 + '0'; // Dịch ngược 5
        }
        // Các ký tự khác giữ nguyên
        s[i] = v3; // Cập nhật ký tự đã xử lý
    }
}
```

```cpp

int main()
{
    string giaima;
    getline(cin, giaima);
    reverse(giaima);
    cout << giaima;
}

```

```
input
Xsfpj
Snake
```

- Kết quả:

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

Bài 3:

Phase 3:

```
v5 = 0;
v4 = 0;
v4 = __isoc99_sscanf(a1, "%d %d", &v3, &v2);
if ( v4 <= 1 )
  explode_bomb();
```

Mục tiêu, nhập vào 2 số v3, v2 thỏa mãn điều kiện của phase 3.

```
v5 -= 292;
if ( v3 > 5 || (result = v2, v5 != v2) )
  explode_bomb();
return result;
```

Điều kiện để bom nổ là: (v3 > 5) hoặc (v5 != v2)

Suy ra, điều kiện để giải phase này là v3 <= 5 và v5 = v2. Nhưng với điều kiện là v3 và v2 phải là những **số nguyên không âm nhỏ nhất**.

Ta thấy phía trên v5 = -292, như vậy cần chọn v3 với **LABEL** phù hợp để tránh phải nhảy đến nhãn explode_bomb().

- Xét trường hợp nhỏ nhất v3 = 0. Khởi đầu chương trình v5 = 0
  - v5 -= 292, sau đó nhảy đến nhãn **LABEL_5**
  - v5 -= 984, sau đó nhảy đến nhãn **LABEL_6**
  - v5 += 766, sau đó nhảy đến nhãn **LABEL_7**
  - v5 -= 364, sau đó nhảy đến nhãn **LABEL_8**
  - v5 += 292, sau đó nhảy đến nhãn **LABEL_9**
  - v5 -= 292, sau đó nhảy đến nhãn **LABEL_10**
  - v5 += 292, sau đó kết thúc switch

  ➔ Suy ra, ta có v5 = -582 – 292 = -874 → **không thể** chọn trường hợp này.
- Xét trường hợp nhỏ nhất v3 = 1. Khởi đầu chương trình v5 = 0
  - v5 -= 984, sau đó nhảy đến nhãn **LABEL_6**
  - v5 += 766, sau đó nhảy đến nhãn **LABEL_7**
  - v5 -= 364, sau đó nhảy đến nhãn **LABEL_8**
  - v5 += 292, sau đó nhảy đến nhãn **LABEL_9**
  - v5 -= 292, sau đó nhảy đến nhãn **LABEL_10**
  - v5 += 292, sau đó kết thúc switch

  ➔ Suy ra, ta có v5 = -290 – 292 = -584 → **không thể** chọn trường hợp này.
- Xét trường hợp nhỏ nhất v3 = 2. Khởi đầu chương trình v5 = 0
  - v5 += 766, sau đó nhảy đến nhãn **LABEL_7**
  - v5 -= 364, sau đó nhảy đến nhãn **LABEL_8**
  - v5 += 292, sau đó nhảy đến nhãn **LABEL_9**
  - v5 -= 292, sau đó nhảy đến nhãn **LABEL_10**
  - v5 += 292, sau đó kết thúc switch

➔ Suy ra, ta có v5 = 694 – 292 = 402 → **có thể** chọn trường hợp này.

- Xét trường hợp nhỏ nhất v3 = 3. Khởi đầu chương trình v5 = 0
  - v5 -= 364, sau đó nhảy đến nhãn **LABEL_8**
  - v5 += 292, sau đó nhảy đến nhãn **LABEL_9**
  - v5 -= 292, sau đó nhảy đến nhãn **LABEL_10**
  - v5 += 292, sau đó kết thúc switch

  ➔ Suy ra, ta có v5 = -72 - 292 = -364 → **không thể** chọn trường hợp này.
- Xét trường hợp nhỏ nhất v3 = 4. Khởi đầu chương trình v5 = 0
  - v5 += 292, sau đó nhảy đến nhãn **LABEL_9**
  - v5 -= 292, sau đó nhảy đến nhãn **LABEL_10**
  - v5 += 292, sau đó kết thúc switch

  ➔ Suy ra, ta có v5 = 292 - 292 = 0 → **có thể** chọn trường hợp này.
- Xét trường hợp nhỏ nhất v3 = 5. Khởi đầu chương trình v5 = 0
  - v5 -= 292, sau đó nhảy đến nhãn **LABEL_10**
  - v5 += 292, sau đó kết thúc switch

  ➔ Suy ra, ta có v5 = 0 - 292 = -292→ nên **không thể** chọn trường hợp này.

➔ Suy cùng, ta có hai cặp (v3;v2) là (2;402) và (4;0) là **thỏa mãn** được yêu cầu của chương trình. Nhưng cặp (4;0) là những số nguyên không âm nhỏ nhất nên sẽ chọn cặp này.

➔ Đáp án cuối cùng là (v3;v2) = (4;0)

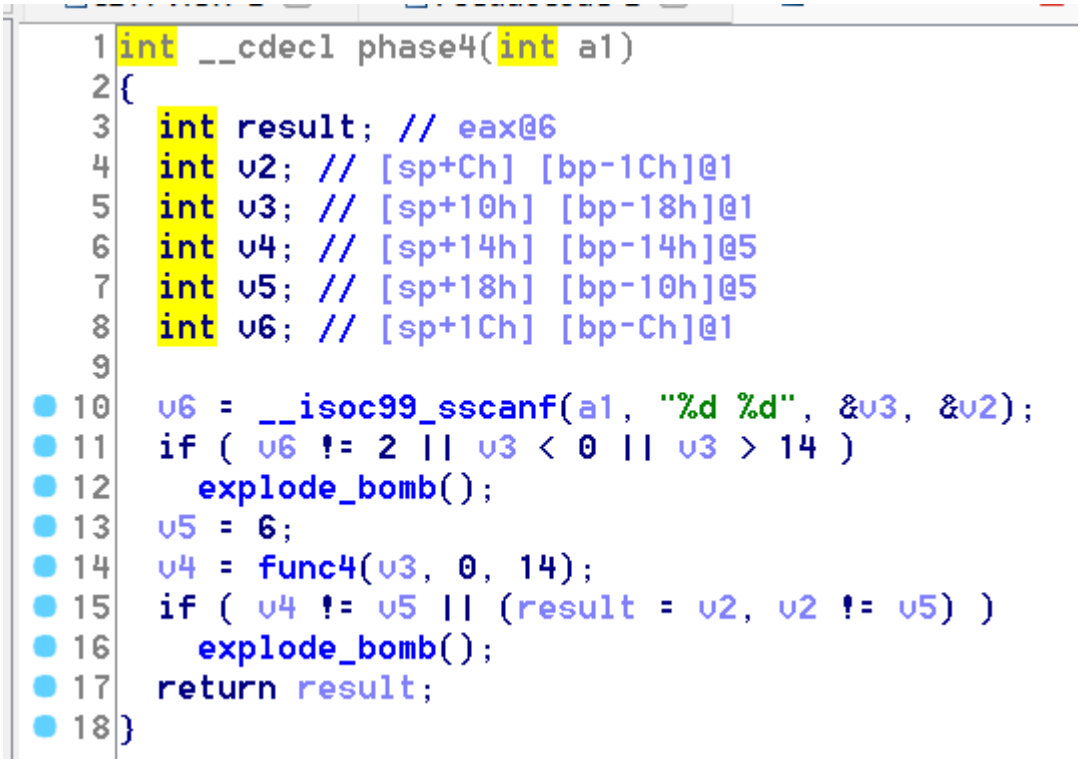
Kết quả:

```
[*] 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 3
- Hint: Many cases make everything so confusing.
2 402
You've beaten another phase, that's great. What about the fourth one?
```

Bài 4:

Phase 4:



```
1  int __cdecl phase4(int a1)
2  {
3    int result; // eax@6
4    int v2; // [sp+Ch] [bp-1Ch]@1
5    int v3; // [sp+10h] [bp-18h]@1
6    int v4; // [sp+14h] [bp-14h]@5
7    int v5; // [sp+18h] [bp-10h]@5
8    int v6; // [sp+1Ch] [bp-Ch]@1
9
10   v6 = __isoc99_sscanf(a1, "%d %d", &v3, &v2);
11   if ( v6 != 2 || v3 < 0 || v3 > 14 )
12     explode_bomb();
13   v5 = 6;
14   v4 = func4(v3, 0, 14);
15   if ( v4 != v5 || (result = v2, v2 != v5) )
16     explode_bomb();
17   return result;
18 }
```

Trong hàm **phase4** yêu cầu ta nhập vào 2 số v3 và v2 cách nhau 1 khoảng trắng.

(**v6 = __isoc99_sscanf(a1, "%d %d", &v3, &v2)**) chuỗi a1 được nhập vào từ bàn phím và đọc bởi hàm sscanf theo định dạng %d %d (**kiểu int**) và lưu vào các biến đích là v3, v2 với cách tham biến, sau đó lưu số giá trị đọc được vào v6.

if (v6 != 2 || v3 < 0 || v3 > 14)

explode_bomb();

Khi một trong các điều kiện trên thỏa thì sẽ gọi hàm explode_bomb(), do đó số lượng phần tử nhập vào (**v6**) phải là 2, **v3>=0 và v3<=14.**

Sau đó biến v4 nhận giá trị trả về từ hàm **fun4** với các đối số truyền vào là v3, 0 và 14.

Khi hàm func4 hoàn tất việc trả về, hàm phase4 kiểm tra các điều kiện một lần nữa trước khi trả về kết quả:

if (v4 != v5 || (result = v2, v2 != v5))

explode_bomb();

return result;

Từ biểu thức điều kiện trong hàm if ta có thể suy ra điều kiện để không gọi hàm **explode_bomb()** là v4 = v5 = 6 và result = v2 = v5 = 6. Vậy giá trị thứ 2 trong 2 giá trị được nhập vào là 6 (v2 = 6), còn giá trị đầu tiên (v3) sẽ tìm được thông qua hàm **func4** sao cho giá trị trả về là 6 (v4=6).

Func4:

```
1  int __cdecl func4(int a1, int a2, int a3)
2  {
3    int result; // eax@2
4    int v4; // [sp+Ch] [bp-Ch]@1
5
6    v4 = (a3 - a2) / 2 + a2;
7    if ( v4 <= a1 )
8    {
9      if ( v4 >= a1 )
10       result = 0;
11     else
12       result = 2 * func4(a1, v4 + 1, a3) + 1;
13   }
14   else
15   {
16     result = 2 * func4(a1, a2, v4 - 1);
17   }
18   return result;
19 }
```

Ta dễ dàng nhận ra được đây là một hàm đệ quy khi tên hàm được gọi lại bên trong chính nó, các tham số của hàm là a1 = v3, a2 = 0, a3 = 14, trong đó chỉ có biến v3 thay đổi theo giá trị được nhập vào, và 0 <= v3 <= 14, do đó ta sẽ thử các trường hợp để tìm ra trường hợp có giá trị trả về **bằng 6** bằng đoạn mã **C++** sau:

Source.cpp

Test

```cpp
#include <iostream>
#include<string.h>
using namespace std;

int 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()
{
    for (int v3 = 0; v3 <= 14; v3++) {
        cout << v3 << "\t";
        cout << func4(v3, 0, 14);
        cout << endl;
    }
    return 0;
}
```

Kết quả C++:

```
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
```

Trong số các trường hợp, chỉ khi v3 = 6 hàm func4 mới trả về kết quả bằng 6. Suy ra số đầu tiên cần nhập vào là 6, cặp số cần tìm là **6 6.**

Kết quả:

```
[*] Phase 4
- Hint: Let's dig in to recursive function :)
6 6
Awesome! Only one phase left!
```

Bài 5:

Phase 5:

```
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_3854[s[i] & 0xF];
    v2 += result;
  }
  if ( v2 != 57 )
    explode_bomb();
  return result;
}
```

Array_3854:

```
.data:0804B200 array_3854      dd 2
.data:0804B204                 db  0Ah
.data:0804B205                 db    0
.data:0804B206                 db    0
.data:0804B207                 db    0
.data:0804B208                 db    6
.data:0804B209                 db    0
.data:0804B20A                 db    0
.data:0804B20B                 db    0
.data:0804B20C                 db    1
.data:0804B20D                 db    0
.data:0804B20E                 db    0
.data:0804B20F                 db    0
.data:0804B210                 db  0Ch
.data:0804B211                 db    0
.data:0804B212                 db    0
.data:0804B213                 db    0
.data:0804B214                 db  10h
.data:0804B215                 db    0
.data:0804B216                 db    0
.data:0804B217                 db    0
.data:0804B218                 db    9
.data:0804B219                 db    0
.data:0804B21A                 db    0
.data:0804B21B                 db    0
.data:0804B21C                 db    3
.data:0804B21D                 db    0
.data:0804B21E                 db    0
.data:0804B21F                 db    0
.data:0804B220                 db    4
.data:0804B221                 db    0
.data:0804B222                 db    0
.data:0804B223                 db    0
.data:0804B224                 db    7
.data:0804B225                 db    0
.data:0804B226                 db    0
.data:0804B227                 db    0
.data:0804B228                 db  0Eh
.data:0804B229                 db    0
.data:0804B22A                 db    0
.data:0804B22B                 db    0
.data:0804B22C                 db    5
.data:0804B22D                 db    0
.data:0804B22E                 db    0
.data:0804B22F                 db    0
.data:0804B230                 db  0Bh
.data:0804B231                 db    0
.data:0804B232                 db    0
.data:0804B233                 db    0
.data:0804B234                 db    8
.data:0804B235                 db    0
.data:0804B236                 db    0
.data:0804B237                 db    0
.data:0804B238                 db  0Fh
.data:0804B239                 db    0
.data:0804B23A                 db    0
.data:0804B23B                 db    0
.data:0804B23C                 db  0Dh
.data:0804B23D                 db    0
.data:0804B23E                 db    0
.data:0804B23F                 db    0
.data:0804B23F _data           ends
.data:0804B23F
```

Kiểu dữ liệu **size_t** dùng để lưu trữ các số nguyên không dấu, thường được dùng để lưu trữ kích thước hay chỉ số của các đối tượng trong chương trình. Ở một số trường hợp, các đặc điểm của **size_t** khá giống với **unsigned int**.

if (result != 6)

explode_bomb();

Sau khi chuẩn bị các biến cục bộ, hàm phase5 kiểm tra độ dài của chuỗi s được truyền vào, nếu độ dài khác 6 sẽ cho bom nổ, vì vậy input mà ta cần nhập vào ở phase 5 là một chuỗi dài 6 kí tự

```
v2 = 0;
for ( i = 0; i <= 5; ++i )
 {
  result = array_3854[s[i] & 0xF];
  v2 += result;
 }
```

Trong vòng for trên, các phần tử trong mảng s sẽ được truy xuất lần lượt và **AND** với **0xF** (0b 0000 1111), nói cách khác, các kí tự ở dạng **binary** tương ứng trong bảng mã **ASCII** sẽ giữ lại 4 bit thấp và bỏ đi 4 bit cao.

Ví dụ: kí tự a= 0b 0110 0001 sau khi **AND** sẽ được 0b 0000 0001.

Giá trị sau khi tính toán sẽ là chỉ số dùng để truy xuất phần tử trong mảng **array_3854**, lúc này, biến v2 sẽ đóng vai trò như biến sum, tính tổng các giá trị từ **array_3854,** vòng for tiếp tục tính toán và truy xuất cho đến khi hết chuỗi s.

```
if ( v2 != 57 )
  explode_bomb();
 return result;
```

Cuối cùng, hàm phase5 so sánh **v2 với 57**, chỉ khi **v2 bằng 57** hàm mới kết thúc mà không cho nổ bom. Do đó, có thể kết luận rằng ta sẽ phải nhập chuỗi 6 kí tự phù hợp sao cho khi lấy 4 bit thấp trong mỗi kí tự làm chỉ số truy xuất trong mảng sẽ trả ra các giá trị có tổng bằng 57. Để chọn ra các kí tự, ta sẽ phân tích **array_3954** và tìm ra 6 giá trị có tổng bằng 57 và suy ra các kí tự.

Mảng **array_3854** có thể biểu diễn như sau trong c++:

Array_3854 [2, 10, 6, 1, 12, 16, 9, 3, 4, 7, 14, 5 11, 8, 15, 13];

Do có nhiều cách để chọn ra 6 số có tổng bằng 57 nên ta cần tìm tổ hợp các giá trị sao cho chỉ số của các giá trị ấy là nhỏ nhất để chuỗi nhập vào là nhỏ nhất-->Ưu tiên chọn các phần tử ở đầu mảng. (Tính theo thứ tự từ trái sang phải của chuỗi kí tự)

Ở vị trí đầu tiên, ta ưu tiên chọn 2 ở vị trí 0, chuỗi cần chọn còn lại 5 giá trị với tổng bằng 55.

Vị trí thứ 2 tiếp tục chọn 2 ở vị trí 0, chuỗi cần chọn còn 4 giá trị với tổng bằng 53.

Nếu vị trí thứ 3 tiếp tục chọn 2 thì sẽ chỉ còn 3 giá trị với tổng bằng 51 --> giá trị trung bình của 3 số còn lại là 17 (51 : 3), mà trong **array_3854**, giá trị lớn nhất chỉ là 16 --> loại. Ta chuyển qua giá trị 10 ở vị trí liền kề, khi này, chuỗi còn 3 kí tự với tổng là 43 --> trung bình 3 số còn lại là 14.33 --> nhận.

Tương tự như trên, nếu vị trí 4 chọn 10, sẽ còn 2 giá trị với trung bình là 16.5 --> loại. Ta tiếp tục thử cho đến giá trị 12 ở vị trí 4, với 11, giá trị trung bình của 2 số còn lại chỉ còn 15.5 --> nhận

Tiếp tục làm như trên cho đến khi đủ 6 giá trị, ta sẽ thu được các giá trị sau:

Chỉ số	0	0	1	4	5	14
Chỉ số binary	0000	0000	0001	0100	0101	1110
Giá trị	2	2	10	12	16	15

Sau đó, ta chọn các kí tự trong bảng **ASCII** có 4 bit thấp bằng với chỉ số **binary**. Vì sẽ có nhiều hơn 1 kí tự phù hợp, ta lại chọn kí tự nhỏ nhất trong số đó. Do yêu cầu của bài lab, các giá trị sẽ được chọn bắt đầu từ kí tự 0, và do chỉ quan tâm đến 4 bit cuối nên chỉ cần 16 kí tự tính từ kí tự 0 là đủ để hoàn thành chuỗi input với giá trị nhỏ nhất:

48	00110000	060	30	0
49	00110001	061	31	1
50	00110010	062	32	2
51	00110011	063	33	3
52	00110100	064	34	4
53	00110101	065	35	5
54	00110110	066	36	6
55	00110111	067	37	7
56	00111000	070	38	8
57	00111001	071	39	9
58	00111010	072	3A	:
59	00111011	073	3B	;
60	00111100	074	3C	<
61	00111101	075	3D	=
62	00111110	076	3E	>
63	00111111	077	3F	?

Căn cứ vào bảng giá trị trên, chuỗi cần nhập vào là **00145>** là đáp án có giá trị nhỏ nhất.

Kết quả:

```
[*] Phase 5
-Hint: No hint is also a hint :)
00145>
Amazing bomb solvers, the bomb has been deactived. Enjoy your day :))
```

Ở đây, ta cũng có thể dùng code **PyThon** để chọn những số có tổng bằng 53 vì đã lấy số hai lần số 2 ở vị trí thứ 0:

```python
from itertools import combinations

# Danh sách đã cung cấp
numbers = [2, 10, 6, 1, 12, 16, 9, 3, 4, 7, 14, 5, 11, 8, 15, 13]

# Tổng đã cho
target_sum = 53

# Tìm 4 phần tử có tổng bằng 53
valid_combinations = [
    (combo, [numbers.index(num) for num in combo])
    for combo in combinations(numbers, 4)
    if sum(combo) == target_sum
]

# In kết quả
print("Valid combinations that sum to 53 along with their indices:")
for combo, indices in valid_combinations:
    print(f"Numbers: {combo}, Indices: {indices}")
```

input

```
Valid combinations that sum to 53 along with their indices:
Numbers: (10, 12, 16, 15), Indices: [1, 4, 5, 14]
Numbers: (10, 16, 14, 13), Indices: [1, 5, 10, 15]
Numbers: (12, 16, 14, 11), Indices: [4, 5, 10, 12]
Numbers: (16, 9, 15, 13), Indices: [5, 6, 14, 15]
Numbers: (16, 14, 8, 15), Indices: [5, 10, 13, 14]
Numbers: (14, 11, 15, 13), Indices: [10, 12, 14, 15]
```

Array_3854 [2, 10, 6, 1, 12, 16, 9, 3, 4, 7, 14, 5 11, 8, 15, 13]

Các kết quả khác:

1. 0_0_1_4_5_14 --> 00145>
2. 0_0_1_5_10_15 --> 0015:?
3. 0_0_4_5_10_12 --> 0045:<
4. 0_0_5_6_14_15 --> 0056>?
5. 0_0_5_10_13_14 --> 005:=>
6. 0_0_10_12_14_15 --> 00:<>?

Tất nhiên, đây chỉ là trường hợp **chỉ số thấp nhất** lấy 2 lần để cho ra **kết quả nhỏ nhất**, còn rất nhiều trường hợp khác nữa để **tổng 6 số bằng 57** mà không tính độ ưu tiên chỉ số.
Sau đây là code **PyThon** để tìm **tất cả các trường hợp** theo độ ưu tiên chỉ số.

```python
# Hàm tìm các tổ hợp thỏa mãn điều kiện
def tim_to_hop(danh_sach, tong_can_tim, to_hop, chi_so, bat_dau, ket_qua):
    # Nếu tổng thỏa mãn và đã chọn đủ 6 số
    if len(to_hop) == 6 and tong_can_tim == 0:
        ket_qua.append((to_hop[:], chi_so[:]))  # Thêm tổ hợp và chỉ số vào kết quả
        return

    # Nếu số lượng vượt quá 6 hoặc tổng âm, dừng đệ quy
    if len(to_hop) > 6 or tong_can_tim < 0:
        return

    # Lặp qua danh sách số, cho phép sử dụng lại số hiện tại
    for i in range(bat_dau, len(danh_sach)):
        to_hop.append(danh_sach[i])
        chi_so.append(i)
        tim_to_hop(danh_sach, tong_can_tim - danh_sach[i], to_hop, chi_so, i, ket_qua)
        to_hop.pop()  # Quay lui
        chi_so.pop()
```

```python
# Danh sách các số cho trước
danh_sach = [2, 10, 6, 1, 12, 16, 9, 3, 4, 7, 14, 5, 11, 8, 15, 13]

# Tổng cần tìm
tong_can_tim = 57

# Biến lưu trữ kết quả
ket_qua = []

# Gọi hàm tìm tổ hợp
tim_to_hop(danh_sach, tong_can_tim, [], [], 0, ket_qua)

# Hàm chuyển đổi chỉ số thành mã ASCII
def chuyen_doi_chi_so(chi_so):
    return ''.join(
        chr(i + 48) if i <= 9 else chr(i + 48)  # Chỉ số từ 0-9 chuyển thành '0'-'9', lớn hơn thành ASCII tương ứng
        for i in chi_so
    )

# In kết quả
print("Các tổ hợp hợp lệ có tổng bằng 57 kèm chỉ số đã chuyển đổi:")
for to_hop, chi_so in ket_qua:
    chi_so_da_chuyen = chuyen_doi_chi_so(chi_so)
    print(f"Tổ hợp: {to_hop}, Chỉ số: {chi_so_da_chuyen}")
```

Sau khi chạy code ta được rất nhiều kết quả nên chỉ có thể lấy tượng trưng một vài kết quả:

Kết quả: