2016년 시스템 프로그래밍

-Bomb Lab-

제출일자	2016.11.14.
이 름	정 윤 수
학 번	201302482
분 반	00

-목 차-

1.첫 번째 폭탄 해설 및 순서도

-어셈블리어 코드

```
0000000000400f90 <phase_1>:
  400f90:   48 83 ec 08          sub    $0x8,%rsp
  400f94:   be a0 26 40 00       mov    $0x4026a0,%esi
  400f99:   e8 3a 04 00 00       callq  4013d8 <strings_not_equal>
  400f9e:   85 c0                test   %eax,%eax
  400fa0:   74 05                je     400fa7 <phase_1+0x17>
  400fa2:   e8 0a 07 00 00       callq  4016b1 <explode_bomb>
  400fa7:   48 83 c4 08          add    $0x8,%rsp
  400fab:   c3                   retq
```

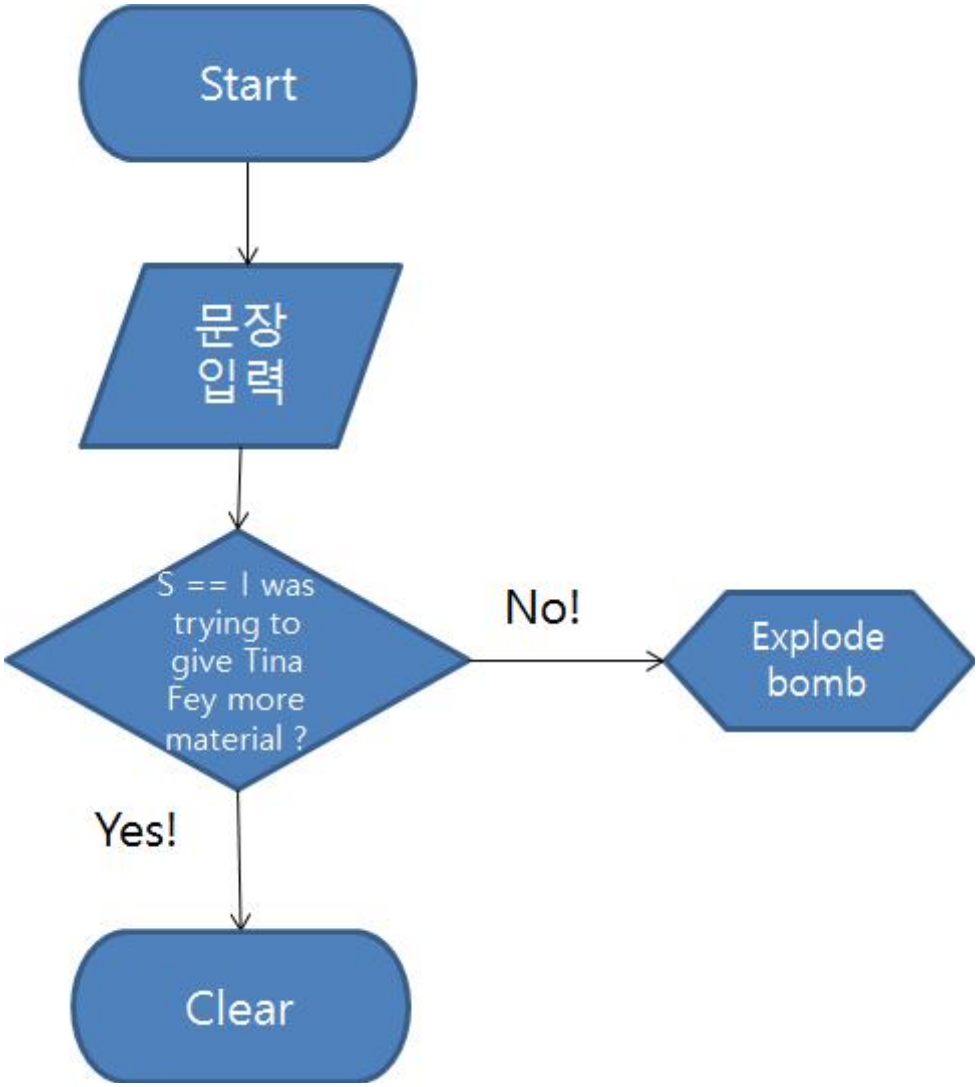
-해결 방법

먼저 phase_1에서 callq명령어로 string_no_equals라는 함수를 호출을 하는 것을 볼수이다. 이 함수는 문장과 문장을 비교를 하는 함수로 두 문자잉 서로 같다면 %eax의 값은 0이 된다. 내가 문장을 입력을 하면 내가 입력을 한 문장은 스택안에 존재 하고 비교를 할려는 문장을 메모리 상에서 가져와 레지스터안에 저장을 할 것이다. 그러므로 함수 호출전에 0x4026a0의 주소를 레지스터에 저장을 하는 것을 보면 0x4026a0에 비교를 하는 문장이 들어있다. x/s 명령어를 이용하여 0x4026a0을 보면 안에 "I was trying to give Tina Fey more material. " 이라는 문장이 들어있는 것을 확인할 수 있다. 입력을 한 문장이 앞에 문장과 같으면 string_not_equal함수는 0의 값을 반환을하여 explode_bomb 함수를 호출을 하는 명령어를 뛰어 넘어 1 번째 폭탄을 클리어할수있게 된다.

-답

"I was trying to give Tina Fey more material. "

- 순서도


Start
문장
입력
S == I was trying to give Tina Fey more material ?
No!
Explode bomb
Yes!
Clear

2.두 번째 폭탄 해설 및 순서도

-어셈블리어 코드

```
0000000000400fac <phase_2>:
  400fac:   55                      push   %rbp
  400fad:   53                      push   %rbx
  400fae:   48 83 ec 28             sub    $0x28,%rsp
  400fb2:   48 89 e6                mov    %rsp,%rsi
  400fb5:   e8 2d 07 00 00          callq  4016e7 <read_six_numbers>
  400fba:   83 3c 24 00             cmpl   $0x0,(%rsp)
  400fbe:   79 24                   jns    400fe4 <phase_2+0x38>
  400fc0:   e8 ec 06 00 00          callq  4016b1 <explode_bomb>
  400fc5:   eb 1d                   jmp    400fe4 <phase_2+0x38>
  400fc7:   89 d8                   mov    %ebx,%eax
  400fc9:   03 45 fc                add    -0x4(%rbp),%eax
  400fcc:   39 45 00                cmp    %eax,0x0(%rbp)
  400fcf:   74 05                   je     400fd6 <phase_2+0x2a>
  400fd1:   e8 db 06 00 00          callq  4016b1 <explode_bomb>
  400fd6:   83 c3 01                add    $0x1,%ebx
  400fd9:   48 83 c5 04             add    $0x4,%rbp
  400fdd:   83 fb 06                cmp    $0x6,%ebx
  400fe0:   75 e5                   jne    400fc7 <phase_2+0x1b>
  400fe2:   eb 0c                   jmp    400ff0 <phase_2+0x44>
  400fe4:   48 8d 6c 24 04          lea    0x4(%rsp),%rbp
  400fe9:   bb 01 00 00 00          mov    $0x1,%ebx
  400fee:   eb d7                   jmp    400fc7 <phase_2+0x1b>
  400ff0:   48 83 c4 28             add    $0x28,%rsp
  400ff4:   5b                      pop    %rbx
  400ff5:   5d                      pop    %rbp
  400ff6:   c3                      retq
```

-해결 방법

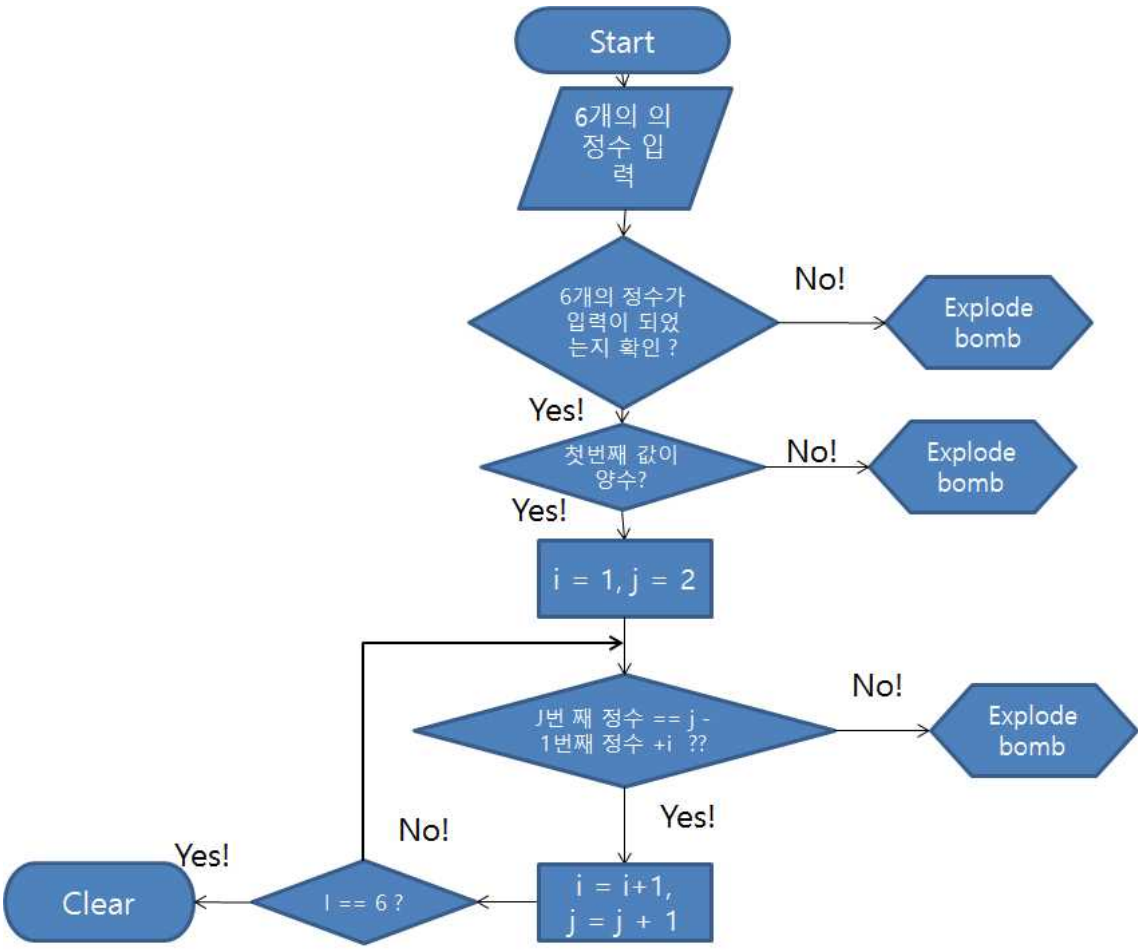
2단계 폭탄에서는 phase_2 함수의 어셈블리어를 보고 폭탄을 해체 하는법을 알 수 있다. 먼저 read_six_numbers 함수를 호출을 하는 것을 보고 6개의 정수값들을 입력을 받는 것을 알 수 있다. read_six_numbers에서는 정수의 입력이 6개가 안되면 explode_bomb함수를 호출하여 폭탄을 터트린다. 올바르게 6개의 정수값을 입력을하면 무사히 read_six_numbers함수를 빠져 나온 후 첫 번째 값이 양수 인지 검사를 하고 양수이면 %rbp에 두 번째 인수의 값을 저장을하고 양수가 아니면 폭탄을 터트린다. 그 후 %rbp와 %ebx를 이용을하여서 2번째 값이 첫 번째 값의 +1값인지 확인을 하고 아니면 폭탄을 터트리고 맞으면 %rbp에 3번째 인수의 값을 저장을 하고 %ebx의 값을 1 증가 시켜 이번에는 세 번째 인수의 값이 2번째 인수의 값의 +2 인지 확인을 한다. 맞으면 다시 4번째 인수를 %rbp에 저장을 하고 %ebx의 값을 1 증가

시켜 주고 다시 확인을 하여 6번 째 인수가 5번째 인수의 +5 가 되는지 까지 모두 확인을 한다. 이 과정을 모두 만족을 한다면 explode_bomb함수를 호출을 하지 않고 폭탄을 해체를 하게 된다.

-답

1 2 4 7 11 16

- 순서도



3.세 번째 폭탄 해설 및 순서도

-어셈블리어 코드

```
0000000000400ff7 <phase_3>:
  400ff7:   48 83 ec 18             sub    $0x18,%rsp
  400ffb:   48 8d 4c 24 0c          lea    0xc(%rsp),%rcx
  401000:   48 8d 54 24 08          lea    0x8(%rsp),%rdx
  401005:   be 9d 29 40 00          mov    $0x40299d,%esi
  40100a:   b8 00 00 00 00          mov    $0x0,%eax
  40100f:   e8 9c fc ff ff          callq  400cb0 <__isoc99_sscanf@plt>
  401014:   83 f8 01                cmp    $0x1,%eax
  401017:   7f 05                   jg     40101e <phase_3+0x27>
  401019:   e8 93 06 00 00          callq  4016b1 <explode_bomb>
  40101e:   83 7c 24 08 07          cmpl   $0x7,0x8(%rsp)
  401023:   77 66                   ja     40108b <phase_3+0x94>
  401025:   8b 44 24 08             mov    0x8(%rsp),%eax
  401029:   ff 24 c5 00 27 40 00    jmpq   *0x402700(,%rax,8)
  401030:   b8 00 00 00 00          mov    $0x0,%eax
  401035:   eb 05                   jmp    40103c <phase_3+0x45>
  401037:   b8 b1 02 00 00          mov    $0x2b1,%eax
  40103c:   2d cd 02 00 00          sub    $0x2cd,%eax
  401041:   eb 05                   jmp    401048 <phase_3+0x51>
  401043:   b8 00 00 00 00          mov    $0x0,%eax
  401048:   05 75 01 00 00          add    $0x175,%eax
  40104d:   eb 05                   jmp    401054 <phase_3+0x5d>
  40104f:   b8 00 00 00 00          mov    $0x0,%eax
  401054:   2d cc 00 00 00          sub    $0xcc,%eax
  401059:   eb 05                   jmp    401060 <phase_3+0x69>
  40105b:   b8 00 00 00 00          mov    $0x0,%eax
  401060:   05 cc 00 00 00          add    $0xcc,%eax
  401065:   eb 05                   jmp    40106c <phase_3+0x75>
  401067:   b8 00 00 00 00          mov    $0x0,%eax
  40106c:   2d cc 00 00 00          sub    $0xcc,%eax
  401071:   eb 05                   jmp    401078 <phase_3+0x81>
  401073:   b8 00 00 00 00          mov    $0x0,%eax
  401078:   05 cc 00 00 00          add    $0xcc,%eax
  40107d:   eb 05                   jmp    401084 <phase_3+0x8d>
  40107f:   b8 00 00 00 00          mov    $0x0,%eax
  401084:   2d cc 00 00 00          sub    $0xcc,%eax
  401089:   eb 0a                   jmp    401095 <phase_3+0x9e>
  40108b:   e8 21 06 00 00          callq  4016b1 <explode_bomb>
```

```
  401090:   b8 00 00 00 00          mov    $0x0,%eax
  401095:   83 7c 24 08 05          cmpl   $0x5,0x8(%rsp)
  40109a:   7f 06                   jg     4010a2 <phase_3+0xab>
  40109c:   3b 44 24 0c             cmp    0xc(%rsp),%eax
  4010a0:   74 05                   je     4010a7 <phase_3+0xb0>
  4010a2:   e8 0a 06 00 00          callq  4016b1 <explode_bomb>
  4010a7:   48 83 c4 18             add    $0x18,%rsp
  4010ab:   c3                      retq
```

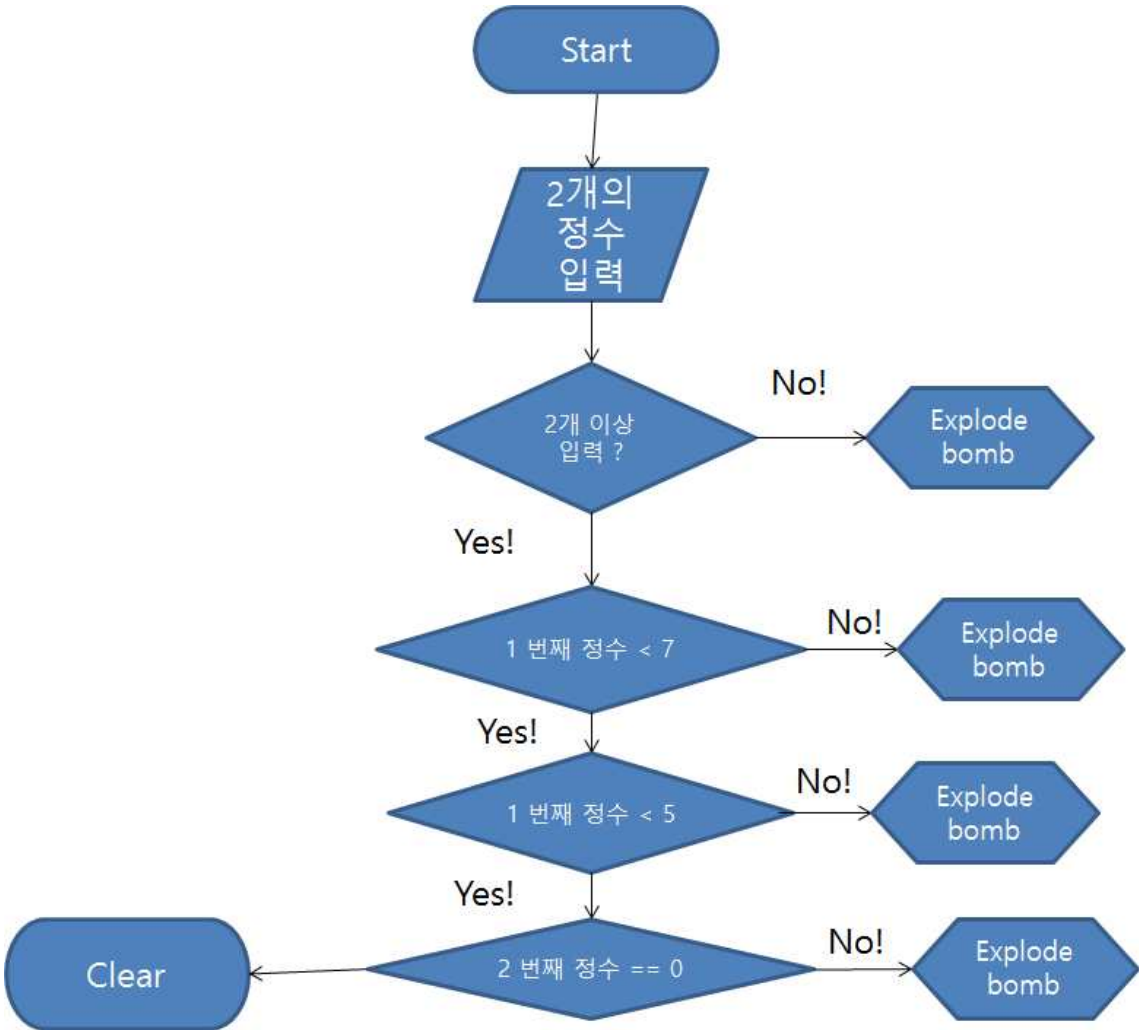
-해결 방법

3번째 폭탄을 해체하기 위해서는 먼저 입력이 어떤 형식으로 되는지 살펴봐
야 한다. 어떤 방식으로 입력이 되는지 살펴보기 위해서는 __isoc99_sscanf함수가 호출을 되는 것을 봐야한다 이 함수는 입력한 값들의 개수를 반환을 해준다 함수 호출 아래에서 반환값이 1보다 커야 한다고 함으로 입력의 개수는 2개 이상이다. %rsp+8의 위치에는 첫 번째로 입력한 값이 들어있다. cmpl명령어로 7과 비교를 하여 값이 7보다 크면 explosive_bomb를 호출을 하는곳으로 점프를 한다. 그러므로 첫 번째 값은 7이하의 값이다. 그 후 %eax로 여러 연산을 하다가 다시 cmpl명령어로 %rsp+8의 값과 5를 비교를 한다. 첫 번째 입력값이 5보다 크면 explosive_bomb 함수를 호출을 함으로 첫 번째 인수는 5보다 작아야 한다. 또한 그 아래에서는 %rsp+12와 %eax의 값을 비교를 하고 있다 %rsp+12에는 현재 두 번째 입력을 한 값이 들어있고 %eax는 0이 저장이 되어있다. 두 값이 서로 같아야지만 폭탄을 피해 갈수 있음으로 두 번째 입력값은 0인 된다.

-해결 방법
4 0

-

순서도


Start
2개의 정수 입력
2개 이상 입력 ?
No!
Explode bomb
Yes!
1 번째 정수 < 7
No!
Explode bomb
Yes!
1 번째 정수 < 5
No!
Explode bomb
Yes!
2 번째 정수 == 0
No!
Explode bomb
Clear

4.네 번째 폭탄 해설 및 순서도

-어셈블리어 코드

```
00000000004010df <phase_4>:
  4010df:   48 83 ec 18          sub    $0x18,%rsp
  4010e3:   48 8d 4c 24 0c       lea    0xc(%rsp),%rcx
  4010e8:   48 8d 54 24 08       lea    0x8(%rsp),%rdx
  4010ed:   be 9d 29 40 00       mov    $0x40299d,%esi
  4010f2:   b8 00 00 00 00       mov    $0x0,%eax
  4010f7:   e8 b4 fb ff ff       callq  400cb0 <__isoc99_sscanf@plt>
  4010fc:   83 f8 02             cmp    $0x2,%eax
  4010ff:   75 07                jne    401108 <phase_4+0x29>
  401101:   83 7c 24 08 0e       cmpl   $0xe,0x8(%rsp)
  401106:   76 05                jbe    40110d <phase_4+0x2e>
  401108:   e8 a4 05 00 00       callq  4016b1 <explode_bomb>
  40110d:   ba 0e 00 00 00       mov    $0xe,%edx
  401112:   be 00 00 00 00       mov    $0x0,%esi
  401117:   8b 7c 24 08          mov    0x8(%rsp),%edi
  40111b:   e8 8c ff ff ff       callq  4010ac <func4>
  401120:   83 f8 07             cmp    $0x7,%eax
  401123:   75 07                jne    40112c <phase_4+0x4d>
  401125:   83 7c 24 0c 07       cmpl   $0x7,0xc(%rsp)
  40112a:   74 05                je     401131 <phase_4+0x52>
  40112c:   e8 80 05 00 00       callq  4016b1 <explode_bomb>
  401131:   48 83 c4 18          add    $0x18,%rsp
  401135:   c3                   retq
```

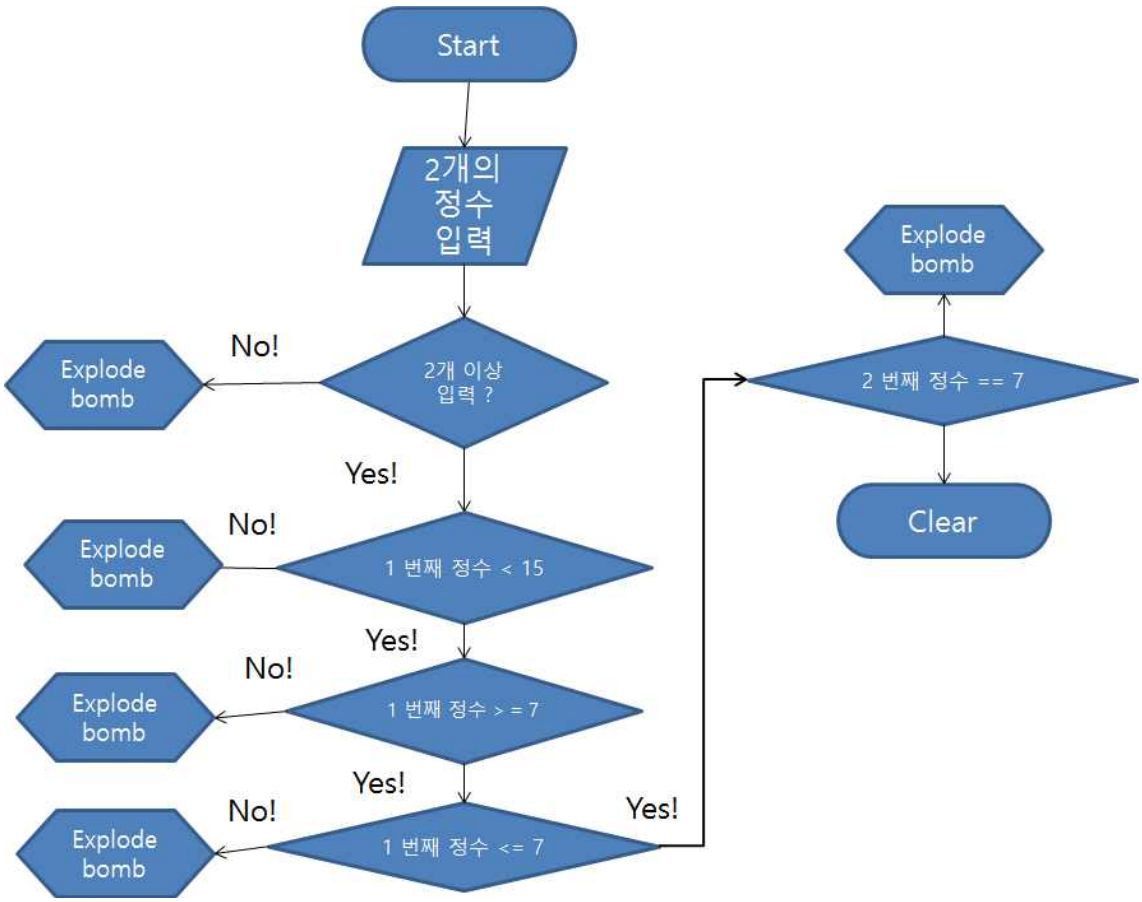
```
00000000004010ac <func4>:
  4010ac:   53                   push   %rbx
  4010ad:   89 d0                mov    %edx,%eax
  4010af:   29 f0                sub    %esi,%eax
  4010b1:   89 c3                mov    %eax,%ebx
  4010b3:   c1 eb 1f             shr    $0x1f,%ebx
  4010b6:   01 d8                add    %ebx,%eax
  4010b8:   d1 f8                sar    %eax
  4010ba:   8d 1c 30             lea    (%rax,%rsi,1),%ebx
  4010bd:   39 fb                cmp    %edi,%ebx
  4010bf:   7e 0c                jle    4010cd <func4+0x21>
  4010c1:   8d 53 ff             lea    -0x1(%rbx),%edx
  4010c4:   e8 e3 ff ff ff       callq  4010ac <func4>
  4010c9:   01 d8                add    %ebx,%eax
  4010cb:   eb 10                jmp    4010dd <func4+0x31>
  4010cd:   89 d8                mov    %ebx,%eax
  4010cf:   39 fb                cmp    %edi,%ebx
  4010d1:   7d 0a                jge    4010dd <func4+0x31>
  4010d3:   8d 73 01             lea    0x1(%rbx),%esi
  4010d6:   e8 d1 ff ff ff       callq  4010ac <func4>
  4010db:   01 d8                add    %ebx,%eax
  4010dd:   5b                   pop    %rbx
  4010de:   c3                   retq
```

-해결 방법

먼저 폭탄을 해체하기 위해서 어떤한 방식으로 입력을 받고 입력의 개수가
몇 개가 되는지를 살펴봐야 한다. 입력을 살펴보기 위해서는 __isoc99_sscanf함수를 살펴보아야 한다. 이 함수는 입력한 값의 개수를 반환을 해준. 반환된 값은 %eax에 저장이 된다 호출 한 후 %eax와 2를 비교를 하는데 입력의 개수가 2개가 아니면 폭탄을 터트린다. 입력의 개수가 올바르면 %rsp+8의 값과 14를 비교를한다. %rsp+8안에는 첫 번째로 입력을 한 값이 저장이 되어있음으로, 첫 번째로 저장이 되어있는 값은 14 이하가 되어야만 폭탄이 터지지 않는다. 그 후 call 명령어를 이용을 하여 func4함수를 호출을 한다. func4 함수에서 여러 연산을 하게 되면 %edi와 %ebx를 비교를 하게 된다 %edi에는 입력을 첫 번째 값이 %ebx에는 7의 값이 저장이 되어있다. 첫 번째 값은 7이하 이어야 하고 점프를 하여 이동을 한 곳에서도 %edi와 %ebx를 비교를 한다. 첫 번째 입력값이 7 이상이어야 func4함수를 빠져 나갈 수 있음으로 이 모든 조건을 만족을하는 값은 7이다. 그러므로 첫 번째 입력값은 7이게 된다. 다시 phase_4함수로 돌아오게 되면 %rsp+12의 값과 7을 비교를 한다. %rsp+12의 주소에는 두 번째 입력값이 저장이 되어있다. 그러므로 두 번째 입력값은 7이다.

-답
7 7

순서도


Start
2개의 정수 입력
2개 이상 입력 ?
No!
Explode bomb
Yes!
1 번째 정수 < 15
No!
Explode bomb
Yes!
1 번째 정수 > = 7
No!
Explode bomb
Yes!
1 번째 정수 <= 7
No!
Explode bomb
Yes!
2 번째 정수 == 7
Explode bomb
Clear

5.다섯 번째 폭탄 해설 및 순서도

-어셈블리어 코드

```
0000000000401136 <phase_5>:
  401136:   53                      push   %rbx
  401137:   48 83 ec 10             sub    $0x10,%rsp
  40113b:   48 89 fb                mov    %rdi,%rbx
  40113e:   64 48 8b 04 25 28 00    mov    %fs:0x28,%rax
  401145:   00 00
  401147:   48 89 44 24 08          mov    %rax,0x8(%rsp)
  40114c:   31 c0                   xor    %eax,%eax
  40114e:   e8 68 02 00 00          callq  4013bb <string_length>
  401153:   83 f8 06                cmp    $0x6,%eax
  401156:   74 42                   je     40119a <phase_5+0x64>
  401158:   e8 54 05 00 00          callq  4016b1 <explode_bomb>
  40115d:   0f 1f 00                nopl   (%rax)
  401160:   eb 38                   jmp    40119a <phase_5+0x64>
  401162:   0f b6 14 03             movzbl (%rbx,%rax,1),%edx
  401166:   83 e2 0f                and    $0xf,%edx
  401169:   0f b6 92 40 27 40 00    movzbl 0x402740(%rdx),%edx
  401170:   88 14 04                mov    %dl,(%rsp,%rax,1)
  401173:   48 83 c0 01             add    $0x1,%rax
  401177:   48 83 f8 06             cmp    $0x6,%rax
  40117b:   75 e5                   jne    401162 <phase_5+0x2c>
  40117d:   c6 44 24 06 00          movb   $0x0,0x6(%rsp)
  401182:   be f6 26 40 00          mov    $0x4026f6,%esi
  401187:   48 89 e7                mov    %rsp,%rdi
  40118a:   e8 49 02 00 00          callq  4013d8 <strings_not_equal>
  40118f:   85 c0                   test   %eax,%eax
  401191:   74 0f                   je     4011a2 <phase_5+0x6c>
  401193:   e8 19 05 00 00          callq  4016b1 <explode_bomb>
  401198:   eb 08                   jmp    4011a2 <phase_5+0x6c>
  40119a:   b8 00 00 00 00          mov    $0x0,%eax
  40119f:   90                      nop
  4011a0:   eb c0                   jmp    401162 <phase_5+0x2c>
  4011a2:   48 8b 44 24 08          mov    0x8(%rsp),%rax
  4011a7:   64 48 33 04 25 28 00    xor    %fs:0x28,%rax
  4011ae:   00 00
  4011b0:   74 05                   je     4011b7 <phase_5+0x81>
  4011b2:   e8 29 fa ff ff          callq  400be0 <__stack_chk_fail@plt>
  4011b7:   48 83 c4 10             add    $0x10,%rsp
  4011bb:   5b                      pop    %rbx
  4011bc:   c3                      retq
```

-'

해결 방법

다섯 번 째 폭탄은 string_length 함수를 호출을 하는 것으로 문장을 입력을 하는 형식이라는 것을 알 수 있다. string_length함수에서는 문장의 길이의 값을 반환을 해주어 %eax안에 저장을한다 함수 호출후에 %eax와 6을 비교하여 서로 같지 않으면 폭탄이 터지게 되어있음으로 문장의 길이는 6이 되어야 한다. 그 후 아래에 strings_not_equals함수가 있는 것이 보인다. 이 함수는 문장들을 비교하여서 서로 같으면 0을 반환을 해주는 함수이다. 비교가 되어지는 문장의 주소를 레지스터 안에 저장을 하고 strings_not_equals 함수를 호출함으로 폭탄을 해체를 하는 문장은 0x4026f6 안에 들어있을 것이다. x/s 명령어로 살펴보면 bruins라는 단어가 들어있다. 하지만 이것을 정답으로 알고 입력을 하면 폭탄이 해체되지 않고 폭탄이 터지게 될 것이다. 그 이유는 strings_not_equals함수를 호출을 하기 전에 그 위에서 스택 안에 문자들을 건드리고 있기 때문이다. 그 이유 때문에 내가 입력을 한 단어는 이상한 단어로 변환이 되어 비교가 되기 때문에 폭탄을 해체를 할 수 없다. 그러므로 내가 입력을 한 단어가 어떻게 변하는지 패턴을 분석을 하여 하나 하나 알파벳의 변하게 된 결과를 구한다. 그러면 내가 입력을 한 문장이 변하여 bruins.가 되게 만들 수 있다.

-답

“mfcdhg. "

순서도


Start
문장
입력
문장의 길이
6 ?
No!
Explode
bomb
Yes!
입력한 문장
== bruins
No!
Explode
bomb
Yes!
Clear

6.여섯 번째 폭탄 해설 및 순서도

-어셈블리어 코드

```
00000000004011bd <phase_6>:
  4011bd:   41 55                    push   %r13
  4011bf:   41 54                    push   %r12
  4011c1:   55                       push   %rbp
  4011c2:   53                       push   %rbx
  4011c3:   48 83 ec 58              sub    $0x58,%rsp
  4011c7:   48 89 e6                 mov    %rsp,%rsi
  4011ca:   e8 18 05 00 00           callq  4016e7 <read_six_numbers>
  4011cf:   49 89 e5                 mov    %rsp,%r13
  4011d2:   41 bc 00 00 00 00        mov    $0x0,%r12d
  4011d8:   4c 89 ed                 mov    %r13,%rbp
  4011db:   41 8b 45 00              mov    0x0(%r13),%eax
  4011df:   83 e8 01                 sub    $0x1,%eax
  4011e2:   83 f8 05                 cmp    $0x5,%eax
  4011e5:   76 05                    jbe    4011ec <phase_6+0x2f>
  4011e7:   e8 c5 04 00 00           callq  4016b1 <explode_bomb>
  4011ec:   41 83 c4 01              add    $0x1,%r12d
  4011f0:   41 83 fc 06              cmp    $0x6,%r12d
  4011f4:   75 07                    jne    4011fd <phase_6+0x40>
  4011f6:   be 00 00 00 00           mov    $0x0,%esi
  4011fb:   eb 42                    jmp    40123f <phase_6+0x82>
  4011fd:   44 89 e3                 mov    %r12d,%ebx
  401200:   48 63 c3                 movslq %ebx,%rax
  401203:   8b 04 84                 mov    (%rsp,%rax,4),%eax
  401206:   39 45 00                 cmp    %eax,0x0(%rbp)
  401209:   75 05                    jne    401210 <phase_6+0x53>
  40120b:   e8 a1 04 00 00           callq  4016b1 <explode_bomb>
  401210:   83 c3 01                 add    $0x1,%ebx
  401213:   83 fb 05                 cmp    $0x5,%ebx
  401216:   7e e8                    jle    401200 <phase_6+0x43>
  401218:   49 83 c5 04              add    $0x4,%r13
  40121c:   eb ba                    jmp    4011d8 <phase_6+0x1b>
  40121e:   48 8b 52 08              mov    0x8(%rdx),%rdx
  401222:   83 c0 01                 add    $0x1,%eax
  401225:   39 c8                    cmp    %ecx,%eax
  401227:   75 f5                    jne    40121e <phase_6+0x61>
  401229:   eb 05                    jmp    401230 <phase_6+0x73>
  40122b:   ba 10 43 60 00           mov    $0x604310,%edx
  401230:   48 89 54 74 20           mov    %rdx,0x20(%rsp,%rsi,2)
  401235:   48 83 c6 04              add    $0x4,%rsi
  401239:   48 83 fe 18              cmp    $0x18,%rsi
```

```
40123d:   74 14                   je     401253 <phase_6+0x96>
40123f:   8b 0c 34                mov    (%rsp,%rsi,1),%ecx
401242:   83 f9 01                cmp    $0x1,%ecx
401245:   7e e4                   jle    40122b <phase_6+0x6e>
401247:   b8 01 00 00 00          mov    $0x1,%eax
40124c:   ba 10 43 60 00          mov    $0x604310,%edx
401251:   eb cb                   jmp    40121e <phase_6+0x61>
401253:   48 8b 5c 24 20          mov    0x20(%rsp),%rbx
401258:   48 8d 44 24 28          lea    0x28(%rsp),%rax
40125d:   48 8d 74 24 50          lea    0x50(%rsp),%rsi
401262:   48 89 d9                mov    %rbx,%rcx
401265:   48 8b 10                mov    (%rax),%rdx
401268:   48 89 51 08             mov    %rdx,0x8(%rcx)
40126c:   48 83 c0 08             add    $0x8,%rax
401270:   48 39 f0                cmp    %rsi,%rax
401273:   74 05                   je     40127a <phase_6+0xbd>
401275:   48 89 d1                mov    %rdx,%rcx
401278:   eb eb                   jmp    401265 <phase_6+0xa8>
40127a:   48 c7 42 08 00 00 00    movq   $0x0,0x8(%rdx)
401281:   00
401282:   bd 05 00 00 00          mov    $0x5,%ebp
401287:   48 8b 43 08             mov    0x8(%rbx),%rax
40128b:   8b 00                   mov    (%rax),%eax
40128d:   39 03                   cmp    %eax,(%rbx)
40128f:   7d 05                   jge    401296 <phase_6+0xd9>
401291:   e8 1b 04 00 00          callq  4016b1 <explode_bomb>
401296:   48 8b 5b 08             mov    0x8(%rbx),%rbx
40129a:   83 ed 01                sub    $0x1,%ebp
40129d:   75 e8                   jne    401287 <phase_6+0xca>
40129f:   48 83 c4 58             add    $0x58,%rsp
4012a3:   5b                      pop    %rbx
4012a4:   5d                      pop    %rbp
4012a5:   41 5c                   pop    %r12
4012a7:   41 5d                   pop    %r13
4012a9:   c3                      retq
```

-해결 방법

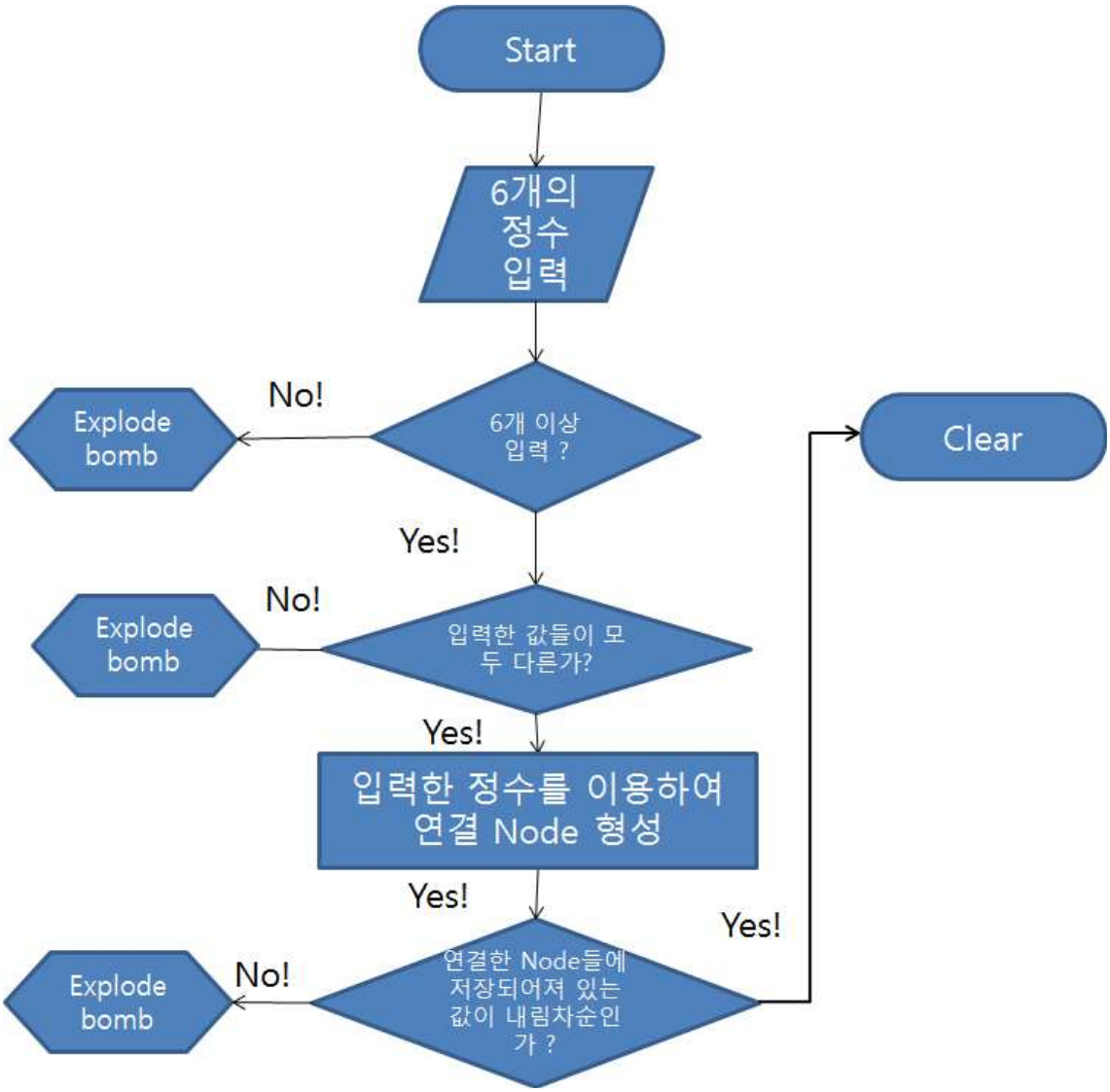
여섯 번째 폭탄은 read_six_numbers함수를 호출을 하는 것으로 보아 6개의 숫자를 입력을 받는다는 것을 알 수 있다. read_six_numbers 함수를 호출을 하면 몇 개의 숫자를 입력을 받았는지 반환을 해준다. 아래에서 반환된 값을 저장을 하여 %eax와 5를 비교를하고 5이하이면 폭탄을 터트리는 것으로 보아 6개 이상의 숫자를 입력하지 않으면 폭탄이 터지는 것을 알 수 있다. 그 후 아래에서는 입력을 한 값들이 서로 모두 같지 않나 하나하나 확인을 하고 만약 같은 같이 있으면 폭탄을 터트린다. 같은 값이 하나도 존재 하지 않는 다면 %rdx에 0x604310의 주소를 저장을 하는 것을 볼 수 있다. 이 주소를 조사를 해보면 0x604310이 node1부터 시작해서 node6까지 이어진 연결

노드 라는 것을 알 수 있다. 입력한 6개의 값과 같은 순서의 연결체인을 형성을 하여 연결된 체인들이 모두 내림차순으로 정렬이 되어져 있는지 확인을 한다. 이 조건을 만족을 하면 폭탄을 해체를 할 수 있다.

-답

3 2 6 1 4 5

순서도



7.숨겨진 폭탄 해설 및 순서도

-어셈블리어 코드

```
0000000000040184f <phase_defused>:
  40184f:   48 83 ec 78             sub    $0x78,%rsp
  401853:   64 48 8b 04 25 28 00    mov    %fs:0x28,%rax
  40185a:   00 00
  40185c:   48 89 44 24 68          mov    %rax,0x68(%rsp)
  401861:   31 c0                   xor    %eax,%eax
  401863:   bf 01 00 00 00          mov    $0x1,%edi
  401868:   e8 38 fd ff ff          callq  4015a5 <send_msg>
  40186d:   83 3d 48 2f 20 00 06    cmpl   $0x6,0x202f48(%rip)        # 60
  401874:   75 6d                   jne    4018e3 <phase_defused+0x94>
  401876:   4c 8d 44 24 10          lea    0x10(%rsp),%r8
  40187b:   48 8d 4c 24 0c          lea    0xc(%rsp),%rcx
  401880:   48 8d 54 24 08          lea    0x8(%rsp),%rdx
  401885:   be e7 29 40 00          mov    $0x4029e7,%esi
  40188a:   bf d0 48 60 00          mov    $0x6048d0,%edi
  40188f:   b8 00 00 00 00          mov    $0x0,%eax
  401894:   e8 17 f4 ff ff          callq  400cb0 <__isoc99_sscanf@plt>
  401899:   83 f8 03                cmp    $0x3,%eax
  40189c:   75 31                   jne    4018cf <phase_defused+0x80>
  40189e:   be f0 29 40 00          mov    $0x4029f0,%esi
  4018a3:   48 8d 7c 24 10          lea    0x10(%rsp),%rdi
  4018a8:   e8 2b fb ff ff          callq  4013d8 <strings_not_equal>
  4018ad:   85 c0                   test   %eax,%eax
  4018af:   75 1e                   jne    4018cf <phase_defused+0x80>
  4018b1:   bf 48 28 40 00          mov    $0x402848,%edi
  4018b6:   e8 05 f3 ff ff          callq  400bc0 <puts@plt>
  4018bb:   bf 70 28 40 00          mov    $0x402870,%edi
  4018c0:   e8 fb f2 ff ff          callq  400bc0 <puts@plt>
  4018c5:   b8 00 00 00 00          mov    $0x0,%eax
  4018ca:   e8 19 fa ff ff          callq  4012e8 <secret_phase>
  4018cf:   bf a8 28 40 00          mov    $0x4028a8,%edi
  4018d4:   e8 e7 f2 ff ff          callq  400bc0 <puts@plt>
  4018d9:   bf d8 28 40 00          mov    $0x4028d8,%edi
  4018de:   e8 dd f2 ff ff          callq  400bc0 <puts@plt>
  4018e3:   48 8b 44 24 68          mov    0x68(%rsp),%rax
  4018e8:   64 48 33 04 25 28 00    xor    %fs:0x28,%rax
  4018ef:   00 00
  4018f1:   74 05                   je     4018f8 <phase_defused+0xa9>
  4018f3:   e8 e8 f2 ff ff          callq  400be0 <__stack_chk_fail@plt>
  4018f8:   48 83 c4 78             add    $0x78,%rsp
  4018fc:   c3                      retq
```

```
00000000004012e8 <secret_phase>:
  4012e8:   53                      push   %rbx
  4012e9:   e8 3b 04 00 00          callq  401729 <read_line>
  4012ee:   ba 0a 00 00 00          mov    $0xa,%edx
  4012f3:   be 00 00 00 00          mov    $0x0,%esi
  4012f8:   48 89 c7                mov    %rax,%rdi
  4012fb:   e8 90 f9 ff ff          callq  400c90 <strtol@plt>
  401300:   48 89 c3                mov    %rax,%rbx
  401303:   8d 40 ff                lea    -0x1(%rax),%eax
  401306:   3d e8 03 00 00          cmp    $0x3e8,%eax

  40130d:   e8 9f 03 00 00          callq  4016b1 <explode_bomb>
  401312:   89 de                   mov    %ebx,%esi
  401314:   bf 30 41 60 00          mov    $0x604130,%edi
  401319:   e8 8c ff ff ff          callq  4012aa <fun7>
  40131e:   83 f8 03                cmp    $0x3,%eax
  401321:   74 05                   je     401328 <secret_phase+0x40>
  401323:   e8 89 03 00 00          callq  4016b1 <explode_bomb>
  401328:   bf d0 26 40 00          mov    $0x4026d0,%edi
  40132d:   e8 8e f8 ff ff          callq  400bc0 <puts@plt>
  401332:   e8 18 05 00 00          callq  40184f <phase_defused>
  401337:   5b                      pop    %rbx
  401338:   c3                      retq
  401339:   0f 1f 80 00 00 00 00    nopl   0x0(%rax)


00000000004012aa <fun7>:
  4012aa:   48 83 ec 08             sub    $0x8,%rsp
  4012ae:   48 85 ff                test   %rdi,%rdi
  4012b1:   74 2b                   je     4012de <fun7+0x34>
  4012b3:   8b 17                   mov    (%rdi),%edx
  4012b5:   39 f2                   cmp    %esi,%edx
  4012b7:   7e 0d                   jle    4012c6 <fun7+0x1c>
  4012b9:   48 8b 7f 08             mov    0x8(%rdi),%rdi
  4012bd:   e8 e8 ff ff ff          callq  4012aa <fun7>
  4012c2:   01 c0                   add    %eax,%eax
  4012c4:   eb 1d                   jmp    4012e3 <fun7+0x39>
  4012c6:   b8 00 00 00 00          mov    $0x0,%eax
  4012cb:   39 f2                   cmp    %esi,%edx
  4012cd:   74 14                   je     4012e3 <fun7+0x39>
  4012cf:   48 8b 7f 10             mov    0x10(%rdi),%rdi
  4012d3:   e8 d2 ff ff ff          callq  4012aa <fun7>
  4012d8:   8d 44 00 01             lea    0x1(%rax,%rax,1),%eax
  4012dc:   eb 05                   jmp    4012e3 <fun7+0x39>
  4012de:   b8 ff ff ff ff          mov    $0xffffffff,%eax
  4012e3:   48 83 c4 08             add    $0x8,%rsp
  4012e7:   c3                      retq
```

-해결 방법

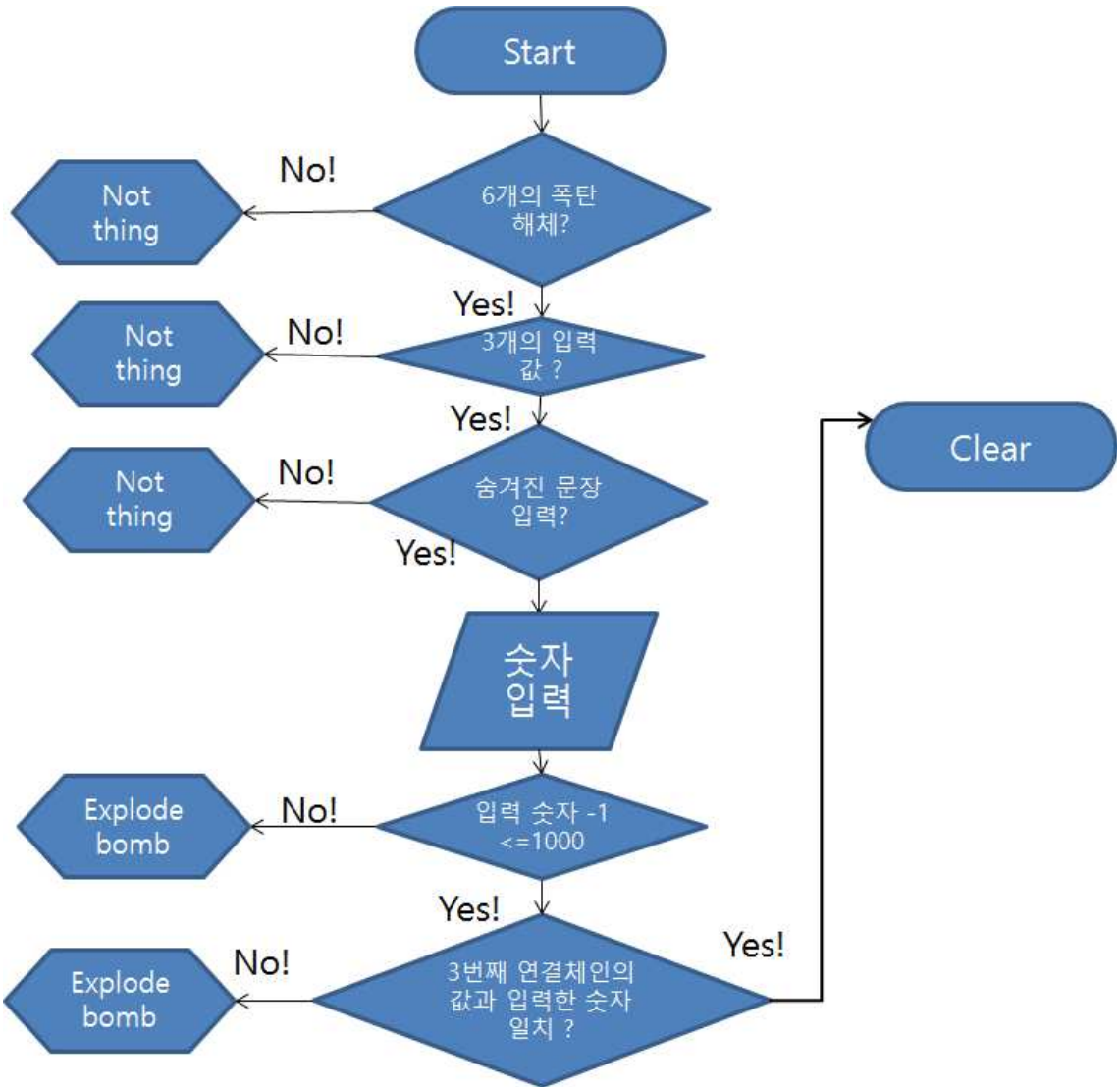
숨겨진 폭탄을 해체하기 위해서는 폭탄을 해체하였을 때 어떤 조건을 만족을 시켜야한다. 1번 째 조건은 6개의 폭탄을 해체를 하는 것이다. 이는 첫 번째 cmpl 명령어에서 확인을 할 수 있다. 0x202f48(%rip)의 값을 확인을 해보면 문제를 풀때마다 1씩 증가를 하게 된다. 6개의 문제를 풀면 이 조건을 클리어 할 수 있다. 2번째 조건은 3개의 입력값이 있어야 한다. 하지만 1번째 조건을 클리어를 한 후에는 문장을 입력을 받을 수 있는 함수는 존재하지 않는다. 하지만 isoc99_sscanf 호출 이전의 0x4029e7의 주소를 조사를 하면 두 개의 정수와 1개의 문장이 인자로 들어간다는 것을 알 수 있다. 이것을 3개의 입력을 해야 한다는 것을 알았지만 입력을 하는 함수가 없다 하지만 계속 폭탄을 해체를 하게 되면 두 개의 정수의 값으로 4번째 폭탄 해체의 값이 들어간다는 것을 알았다. 3번째 조건으로는 부족한 하나의 문장을 채워 넣는 것이다. 이 문장은 strings_not_equals함수로 조사를 하게 될것임으로 함수를 호출하기 이전에 0x4029f0주소를 조사를 하게 되면 "DrEvil"이라는 단어가 있는 것을 볼 수 있다. 이것들을 모두 총합하면 6번째 폭탄 까지 모두 풀면서 4번째 답으로 입력을 3개를 하면서 3번째 입력이 "DrEvil"이 되어야 한다. 이 조건들을 충족을 시키면 숨겨진 폭탄을 해체를 할 수 있는 문제가 나온다. secret_phase에서는 입력을 하나 받는다. 입력을 받고난후 cmpl 명령어로 입력 값 -1의 값이 1000 이하 이어야만 한다는 것을 알 수 있다. 그러므로 최대값으로 1001을 입력을 할 수 있다. 그 후 아래를 보면 fun7으로 얻은 반환값이 3 이어야만 폭탄을 해체를 할수있다는 것을 볼 수 있다. fun7에서는 연결체인을 사용을 한다 %rdi 에 연결체인의 주소가 저장이 되어있어 반환값으로 3을 얻기 위해서는 3번째 연결체인의 값이랑 입력을 한 값이랑 서로 일치를 해야한다. 내가 입력을 한 값과 3번째 연결체인의 값이 일치한다면 숨겨진 폭탄을 해체를 할 수 있다.

-답

"4번 폭탄 답 : 7 7 DrEvil"

"숨겨진 폭탄 답 : 107 "

순서도


Start
6개의 폭탄 해체?
No!
Not thing
Yes!
3개의 입력 값 ?
No!
Not thing
Yes!
숨겨진 문장 입력?
No!
Not thing
Yes!
숫자 입력
입력 숫자 -1 <=1000
No!
Explode bomb
Yes!
3번째 연결체인의 값과 입력한 숫자 일치 ?
No!
Explode bomb
Yes!
Clear

-고찰 및 느낀점-

3 bomb3	Wed Nov 9 16:30	7	0	70	valid

```
Starting program: /home/sys00/a201302482/bomb3/bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
I was trying to give Tina Fey more material.
Phase 1 defused. How about the next one?
1 2 4 7 11 16
That's number 2.  Keep going!
4 0
Halfway there!
7 7 DrEvil
So you got that one.  Try this one.
mfcdhg
Good work!  On to the next...
3 2 6 1 4 5
Curses, you've found the secret phase!
But finding it and solving it are quite different...
107
Wow! You've defused the secret stage!
Congratulations! You've defused the bomb!
Your instructor has been notified and will verify your solution.
```

다른 사람들과 순위를 다투면서 문제를 푸는 것이 다른 사람에게 지고 싶지 않다는 경쟁심과 누구 보다 더 빨리 풀고자하는 마음을 자극을 하여 매우 재미있었다. 또한 어셈블리어에 대해서도 좀 더 궁리를 하면서 여러 가지 생각을 해볼수 있었고 답을 몇시간에 걸쳐 하나 하나 풀릴 때 성취감을 느낄 수 있었다.