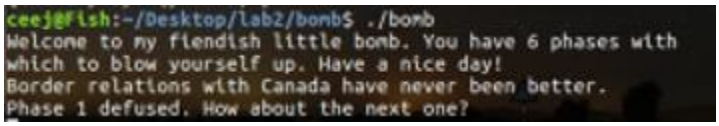
LAB2

Phase_1

```
0000000000400ee0 <phase_1>:
   400ee0:    48 83 ec 08          sub    $0x8,%rsp
   400ee4:    be 00 24 40 00       mov    $0x402400,%esi
   400ee9:    e8 4a 04 00 00       callq  401338 <strings_not_equal>
   400eee:    85 c0                test   %eax,%eax
   400ef0:    74 05                je     400ef7 <phase_1+0x17>
   400ef2:    e8 43 05 00 00       callq  40143a <explode_bomb>
   400ef7:    48 83 c4 08          add    $0x8,%rsp
   400efb:    c3                   retq
```

猜测%esi 寄存器中的值就是<strings_not_equal>中比较字符串的地址，所以在 gdb 中设置 break phase_1，stepi 到 0x400ee9 后 x/1sb 得到存在 0x402400 地址后的字符串为"Border relations with Canada have never been better."。

输入测试成功：



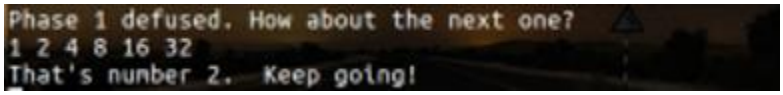
Phase_2

```
0000000000400efc <phase_2>:
   400efc:55                       push   %rbp
   400efd:    53                   push   %rbx
   400efe:    48 83 ec 28          sub    $0x28,%rsp
   400f02:    48 89 e6             mov    %rsp,%rsi
   400f05:    e8 52 05 00 00       callq  40145c <read_six_numbers>
   400f0a:    83 3c 24 01          cmpl   $0x1,(%rsp)
   400f0e:    74 20                je     400f30 <phase_2+0x34>
   400f10:    e8 25 05 00 00       callq  40143a <explode_bomb>
   400f15:    eb 19                jmp    400f30 <phase_2+0x34>
   400f17:    8b 43 fc             mov    -0x4(%rbx),%eax
   400f1a:    01 c0                add    %eax,%eax
   400f1c:    39 03                cmp    %eax,(%rbx)
   400f1e:    74 05                je     400f25 <phase_2+0x29>
   400f20:    e8 15 05 00 00       callq  40143a <explode_bomb>
   400f25:    48 83 c3 04          add    $0x4,%rbx
   400f29:    48 39 eb             cmp    %rbp,%rbx
   400f2c:    75 e9                jne    400f17 <phase_2+0x1b>
   400f2e:    eb 0c                jmp    400f3c <phase_2+0x40>
   400f30:    48 8d 5c 24 04       lea    0x4(%rsp),%rbx
   400f35:    48 8d 6c 24 18       lea    0x18(%rsp),%rbp
   400f3a:    eb db                jmp    400f17 <phase_2+0x1b>
   400f3c:    48 83 c4 28          add    $0x28,%rsp
   400f40:    5b                   pop    %rbx
```

```
400f41:	5d	pop	%rbp
400f42:	c3	retq
```

读汇编语言后发现在<read_six_numbers>之后第一个先将 1 与第一个数字比较，若相等则跳到 0x400F30 进行第二个数字的读入再跳到 0X400F17，将%eax 置为第一个读的数字，将两倍的%eax 和第二个读入的数字进行比较，如果相同则跳到 0X400F25，进入重复的回到 0X400F17 的循环，一共四次，因此可以知道六个数字分别为“1 2 4 8 16 32”

测试结果如下：



Phase_3

```
0000000000400f43 <phase_3>:
400f43:	48 83 ec 18	sub	$0x18,%rsp
400f47:	48 8d 4c 24 0c	lea	0xc(%rsp),%rcx
400f4c:	48 8d 54 24 08	lea	0x8(%rsp),%rdx
400f51:	be cf 25 40 00	mov	$0x4025cf,%esi
400f56:	b8 00 00 00 00	mov	$0x0,%eax
400f5b:	e8 90 fc ff ff	callq	400bf0 <__isoc99_sscanf@plt>
400f60:	83 f8 01	cmp	$0x1,%eax
400f63:	7f 05	jg	400f6a <phase_3+0x27>
400f65:	e8 d0 04 00 00	callq	40143a <explode_bomb>
400f6a:	83 7c 24 08 07	cmpl	$0x7,0x8(%rsp)
400f6f:77 3c	ja	400fad <phase_3+0x6a>
400f71:	8b 44 24 08	mov	0x8(%rsp),%eax
400f75:	ff 24 c5 70 24 40 00	jmpq	*0x402470(,%rax,8)
400f7c:	b8 cf 00 00 00	mov	$0xcf,%eax
400f81:	eb 3b	jmp	400fbe <phase_3+0x7b>
400f83:	b8 c3 02 00 00	mov	$0x2c3,%eax
400f88:	eb 34	jmp	400fbe <phase_3+0x7b>
400f8a:	b8 00 01 00 00	mov	$0x100,%eax
400f8f:eb 2d	jmp	400fbe <phase_3+0x7b>
400f91:	b8 85 01 00 00	mov	$0x185,%eax
400f96:	eb 26	jmp	400fbe <phase_3+0x7b>
400f98:	b8 ce 00 00 00	mov	$0xce,%eax
400f9d:	eb 1f	jmp	400fbe <phase_3+0x7b>
400f9f:b8 aa 02 00 00	mov	$0x2aa,%eax
400fa4:	eb 18	jmp	400fbe <phase_3+0x7b>
400fa6:	b8 47 01 00 00	mov	$0x147,%eax
400fab:	eb 11	jmp	400fbe <phase_3+0x7b>
400fad:	e8 88 04 00 00	callq	40143a <explode_bomb>
400fb2:	b8 00 00 00 00	mov	$0x0,%eax
400fb7:	eb 05	jmp	400fbe <phase_3+0x7b>
400fb9:	b8 37 01 00 00	mov	$0x137,%eax
```

```
400fbe:	3b 44 24 0c	cmp	0xc(%rsp),%eax
400fc2:	74 05	je	400fc9 <phase_3+0x86>
400fc4:	e8 71 04 00 00	callq	40143a <explode_bomb>
400fc9:	48 83 c4 18	add	$0x18,%rsp
400fcd:	c3	retq
```

在 0x400f6a 处我们得知第一个输入数据一定要小于等于 7 然后 jmp *0x402470(,%rax,8)，%rax 为第一个输入的数据，于是利用 gdb 得到对应表格：

序号：地址：存储值：需要的 HEX：DEC

0：0x402470：0x400f7c：0xCF：207

1：0x402478：0x400fb9：0x137：311

2：0x402480：0x400f83：0x2C3：707

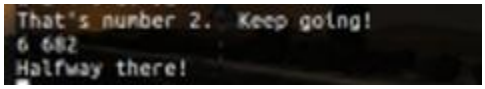
3：0x402488：0x400f8a：0x100：256

4：0x402490：0x400f91：0x185：389

5：0x402498：0x400f98：0xCE：206

6：0x4024a0：0x400f9f： 0x2AA：682

7：0x4024a8：0x400fa6：0x147：327　其中任意一组（序号,DEC）数据都能通过：




```
0000000000400fce <func4>:
400fce:48 83 ec 08	sub	$0x8,%rsp
400fd2:	89 d0	mov	%edx,%eax
400fd4:	29 f0	sub	%esi,%eax
400fd6:	89 c1	mov	%eax,%ecx
400fd8:	c1 e9 1f	shr	$0x1f,%ecx
400fdb:	01 c8	add	%ecx,%eax
400fdd:	d1 f8	sar	%eax
400fdf:8d 0c 30	lea	(%rax,%rsi,1),%ecx
400fe2:	39 f9	cmp	%edi,%ecx
400fe4:	7e 0c	jle	400ff2 <func4+0x24>
400fe6:	8d 51 ff	lea	-0x1(%rcx),%edx
400fe9:	e8 e0 ff ff ff	callq	400fce <func4>
400fee:	01 c0	add	%eax,%eax
400ff0:eb 15	jmp	401007 <func4+0x39>
400ff2:b8 00 00 00 00	mov	$0x0,%eax
400ff7:39 f9	cmp	%edi,%ecx
400ff9:7d 0c	jge	401007 <func4+0x39>
400ffb:8d 71 01	lea	0x1(%rcx),%esi
400ffe:e8 cb ff ff ff	callq	400fce <func4>
401003:	8d 44 00 01	lea	0x1(%rax,%rax,1),%eax
401007:	48 83 c4 08	add	$0x8,%rsp
40100b:	c3	retq
```

Phase_4

```
000000000040100c <phase_4>:
   40100c:	48 83 ec 18		sub	$0x18,%rsp
   401010:	48 8d 4c 24 0c		lea	0xc(%rsp),%rcx
   401015:	48 8d 54 24 08		lea	0x8(%rsp),%rdx
   40101a:	be cf 25 40 00		mov	$0x4025cf,%esi
   40101f:	b8 00 00 00 00		mov	$0x0,%eax
   401024:	e8 c7 fb ff ff		callq	400bf0 <__isoc99_sscanf@plt>
   401029:	83 f8 02		cmp	$0x2,%eax
   40102c:	75 07		jne	401035 <phase_4+0x29>
   40102e:	83 7c 24 08 0e		cmpl	$0xe,0x8(%rsp)
   401033:	76 05		jbe	40103a <phase_4+0x2e>
   401035:	e8 00 04 00 00		callq	40143a <explode_bomb>
   40103a:	ba 0e 00 00 00		mov	$0xe,%edx
   40103f:	be 00 00 00 00		mov	$0x0,%esi
   401044:	8b 7c 24 08		mov	0x8(%rsp),%edi
   401048:	e8 81 ff ff ff		callq	400fce <func4>
   40104d:	85 c0		test	%eax,%eax
   40104f:	75 07		jne	401058 <phase_4+0x4c>
   401051:	83 7c 24 0c 00		cmpl	$0x0,0xc(%rsp)
   401056:	74 05		je	40105d <phase_4+0x51>
   401058:	e8 dd 03 00 00		callq	40143a <explode_bomb>
   40105d:	48 83 c4 18		add	$0x18,%rsp
   401061:	c3		retq
```

从 0x401029 得出必须输入两个数据，从 0x40104d 得出 func4 的返回值必须为 0。通过 0x401051 得出第二个输入的值必须为 0.现在来看 func4，还原得到 func4 为：

```
Int func4(int a,int b,int c){
	Int ret=a/2;
	Int tmp=a/2+b;
	If(c<tmp){
		ret=2*func4(a-1,b,c);
		Return ret;
	}
	ret=0;
	If(c>0){
		Return 2*fun4(a,b+1,c)+1;
	}
	Return ret;
}
```

其中 a 为 14，b 为 0，c 为第一个输入的数字，通过分析递归 func4 发现如果 c>0orc<0 都不能得到 0，所以 c=0。测试结果如下：

Phase_5

```
0000000000401062 <phase_5>:
   401062:    53                          push   %rbx
   401063:    48 83 ec 20                 sub    $0x20,%rsp
   401067:    48 89 fb                    mov    %rdi,%rbx
   40106a:    64 48 8b 04 25 28 00        mov    %fs:0x28,%rax
   401071:    00 00
   401073:    48 89 44 24 18              mov    %rax,0x18(%rsp)
   401078:    31 c0                       xor    %eax,%eax
   40107a:    e8 9c 02 00 00              callq  40131b <string_length>
   40107f:    83 f8 06                    cmp    $0x6,%eax
   401082:    74 4e                       je     4010d2 <phase_5+0x70>
   401084:    e8 b1 03 00 00              callq  40143a <explode_bomb>
   401089:    eb 47                       jmp    4010d2 <phase_5+0x70>
   40108b:    0f b6 0c 03                 movzbl (%rbx,%rax,1),%ecx
   40108f:    88 0c 24                    mov    %cl,(%rsp)
   401092:    48 8b 14 24                 mov    (%rsp),%rdx
   401096:    83 e2 0f                    and    $0xf,%edx
   401099:    0f b6 92 b0 24 40 00        movzbl 0x4024b0(%rdx),%edx
   4010a0:    88 54 04 10                 mov    %dl,0x10(%rsp,%rax,1)
   4010a4:    48 83 c0 01                 add    $0x1,%rax
   4010a8:    48 83 f8 06                 cmp    $0x6,%rax
   4010ac:    75 dd                       jne    40108b <phase_5+0x29>
   4010ae:    c6 44 24 16 00              movb   $0x0,0x16(%rsp)
   4010b3:    be 5e 24 40 00              mov    $0x40245e,%esi
   4010b8:    48 8d 7c 24 10              lea    0x10(%rsp),%rdi
   4010bd:    e8 76 02 00 00              callq  401338 <strings_not_equal>
   4010c2:    85 c0                       test   %eax,%eax
   4010c4:    74 13                       je     4010d9 <phase_5+0x77>
   4010c6:    e8 6f 03 00 00              callq  40143a <explode_bomb>
   4010cb:    0f 1f 44 00 00              nopl   0x0(%rax,%rax,1)
   4010d0:    eb 07                       jmp    4010d9 <phase_5+0x77>
   4010d2:    b8 00 00 00 00              mov    $0x0,%eax
   4010d7:    eb b2                       jmp    40108b <phase_5+0x29>
   4010d9:    48 8b 44 24 18              mov    0x18(%rsp),%rax
   4010de:    64 48 33 04 25 28 00        xor    %fs:0x28,%rax
   4010e5:    00 00
   4010e7:    74 05                       je     4010ee <phase_5+0x8c>
   4010e9:    e8 42 fa ff ff              callq  400b30 <__stack_chk_fail@plt>
   4010ee:    48 83 c4 20                 add    $0x20,%rsp
   4010f2:    5b                          pop    %rbx
   4010f3:    c3                          retq
```

程序首先读入了 6 个字符，核心代码为 0x40108b 到 0x4010ac 的一个 for 循环，（首先$eax=0）

movzbl (%rbx,%rax,1),%ecx　　//ecx=rbx+1*rax

```
mov      %cl,(%rsp)                //rsp 指向 ecx 代表的字符串
mov      (%rsp),%rdx               //rdx 指向 ecx 代表的字符串
and      $0xf,%edx                 //将 exd 保留后 4 位
movzbl 0x4024b0(%rdx),%edx         //edx+=0x4024b0
mov      %dl,0x10(%rsp,%rax,1)//暂存在 rsp+1*rax+0x10 中
add      $0x1,%rax
cmp      $0x6,%rax
jne      40108b <phase_5+0x29>
```

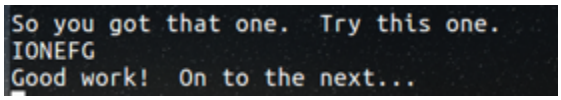
实现了将读入的六个字符取后 4 位进行地址变换得到另外六个字符的操作，地址变化的基础时 0x40245e 中存储的字符串，我们利用 x/s 0x4024b0 得到：

0x4024b0 <array.3449>: "maduiersnfotvbylSo you think you can stop the bomb with ctrl-c, do you?"

利用(gdb) x/s 0x40245e 得到最后比较的字符串 0x40245e: "flyers"

F(9)l(15)y(14)e(5)r(6)(7)输入的 ascii 码的值的十六进制第二位满足 9fe567 就行了

输入“IONEFG”测试成功：



Phase_6

00000000004010f4 <phase_6>:

```
4010f4:    41 56                push    %r14
4010f6:    41 55                push    %r13
4010f8:    41 54                push    %r12
4010fa:    55                   push    %rbp
4010fb:    53                   push    %rbx
4010fc:    48 83 ec 50          sub     $0x50,%rsp
401100:    49 89 e5             mov     %rsp,%r13
401103:    48 89 e6             mov     %rsp,%rsi
401106:    e8 51 03 00 00       callq   40145c <read_six_numbers>//读入 6 个数字
40110b:    49 89 e6             mov     %rsp,%r14
40110e:    41 bc 00 00 00 00    mov     $0x0,%r12d
401114:    4c 89 ed             mov     %r13,%rbp
401117:    41 8b 45 00          mov     0x0(%r13),%eax
40111b:    83 e8 01             sub     $0x1,%eax
40111e:    83 f8 05             cmp     $0x5,%eax//所有数字小于等于 6
401121:    76 05                jbe     401128 <phase_6+0x34>
401123:    e8 12 03 00 00       callq   40143a <explode_bomb>
401128:    41 83 c4 01          add     $0x1,%r12d
40112c:    41 83 fc 06          cmp     $0x6,%r12d
401130:    74 21                je      401153 <phase_6+0x5f>
401132:    44 89 e3             mov     %r12d,%ebx//循环计数器，循环 6 次
401135:    48 63 c3             movslq %ebx,%rax
401138:    8b 04 84             mov     (%rsp,%rax,4),%eax
```

```
40113b:	39 45 00	cmp	%eax,0x0(%rbp)//判断非 0
40113e:	75 05	jne	401145 <phase_6+0x51>
401140:	e8 f5 02 00 00	callq	40143a <explode_bomb>
401145:	83 c3 01	add	$0x1,%ebx
401148:	83 fb 05	cmp	$0x5,%ebx
40114b:	7e e8	jle	401135 <phase_6+0x41>//循环尾部
40114d:	49 83 c5 04	add	$0x4,%r13
401151:	eb c1	jmp	401114 <phase_6+0x20>
401153:	48 8d 74 24 18	lea	0x18(%rsp),%rsi//循环改变输入元素的值为
7-input
401158:	4c 89 f0	mov	%r14,%rax
40115b:	b9 07 00 00 00	mov	$0x7,%ecx
401160:	89 ca	mov	%ecx,%edx
401162:	2b 10	sub	(%rax),%edx
401164:	89 10	mov	%edx,(%rax)
401166:	48 83 c0 04	add	$0x4,%rax
40116a:	48 39 f0	cmp	%rsi,%rax
40116d:	75 f1	jne	401160 <phase_6+0x6c>//循环尾

40116f:	be 00 00 00 00	mov	$0x0,%esi
401174:	eb 21	jmp	401197 <phase_6+0xa3>


401176:	48 8b 52 08	mov	0x8(%rdx),%rdx
40117a:	83 c0 01	add	$0x1,%eax
40117d:	39 c8	cmp	%ecx,%eax
40117f:	75 f5	jne	401176 <phase_6+0x82>
401181:	eb 05	jmp	401188 <phase_6+0x94>


401183:	ba d0 32 60 00	mov	$0x6032d0,%edx
401188:	48 89 54 74 20	mov	%rdx,0x20(%rsp,%rsi,2)//放入 Node
40118d:	48 83 c6 04	add	$0x4,%rsi
401191:	48 83 fe 18	cmp	$0x18,%rsi
401195:	74 14	je	4011ab <phase_6+0xb7>


401197:	8b 0c 34	mov	(%rsp,%rsi,1),%ecx
40119a:	83 f9 01	cmp	$0x1,%ecx	//if a[i]<=1
40119d:	7e e4	jle	401183 <phase_6+0x8f>
40119f:	b8 01 00 00 00	mov	$0x1,%eax
4011a4:	ba d0 32 60 00	mov	$0x6032d0,%edx
4011a9:	eb cb	jmp	401176 <phase_6+0x82>
4011ab:	48 8b 5c 24 20	mov	0x20(%rsp),%rbx//链表建立的循环
```

```
4011b0:	48 8d 44 24 28		lea	0x28(%rsp),%rax
4011b5:	48 8d 74 24 50		lea	0x50(%rsp),%rsi
4011ba:	48 89 d9		mov	%rbx,%rcx
4011bd:	48 8b 10		mov	(%rax),%rdx
4011c0:	48 89 51 08		mov	%rdx,0x8(%rcx)/将下一个 Node 的地址复制给前一个
4011c4:	48 83 c0 08		add	$0x8,%rax
4011c8:	48 39 f0		cmp	%rsi,%rax
4011cb:	74 05			je	4011d2 <phase_6+0xde>
4011cd:	48 89 d1		mov	%rdx,%rcx
4011d0:	eb eb			jmp	4011bd <phase_6+0xc9>//循环尾部
4011d2:	48 c7 42 08 00 00 00	movq	$0x0,0x8(%rdx)//将最后一个 Node 的指针变成 NULL
4011d9:	00
4011da:	bd 05 00 00 00		mov	$0x5,%ebp	//计数器
4011df:	48 8b 43 08		mov	0x8(%rbx),%rax
4011e3:	8b 00			mov	(%rax),%eax
4011e5:	39 03			cmp	%eax,(%rbx)//前一个链表存的数字大于等于后一个？
4011e7:	7d 05			jge	4011ee <phase_6+0xfa>
4011e9:	e8 4c 02 00 00		callq	40143a <explode_bomb>
4011ee:	48 8b 5b 08		mov	0x8(%rbx),%rbx//指针指向下一个
4011f2:	83 ed 01		sub	$0x1,%ebp	//计步器减去 1
4011f5:	75 e8			jne	4011df <phase_6+0xeb>//循环尾部
4011f7:	48 83 c4 50		add	$0x50,%rsp
4011fb:	5b			pop	%rbx
4011fc:	5d			pop	%rbp
4011fd:	41 5c			pop	%r12
4011ff:	41 5d			pop	%r13
401201:	41 5e			pop	%r14
401203:	c3			retq
```

参照代码中的注释，Phase_5 是读入 6 个输入，必须是（123456）的排列，在变成 7-input[i]，再按照这个数组将一组数据编成链表，最后比较链表中数据是否递减，如果是的话则成功。

通过(gdb) x/24dw 0x6032d0 得到：

```
0x6032d0 <node1>:	332  1	6304480  0
0x6032e0 <node2>:	168  2	6304496  0
0x6032f0 <node3>:	924  3	6304512  0
0x603300 <node4>:	691  4	6304528  0
0x603310 <node5>:	477  5	6304544  0
```

0x603320 <node6>:　　443　6　　0　　0

所以从大到小得到顺序为”3 4 5 6 1 2”，因为有 7-input[i]的过程，所以变成”4 3 2 1 6 5”,测试结果如下：