首先为了方便查看代码，我利用objdump -d，将反汇编代码输出到文本文件中。

Main（），查看主函数，知道是输入一个字符串，如果不相符就会爆炸，有六个炸弹和隐藏炸弹。

phase\_1：首先是将栈顶指针减8，接着就是进栈字符串，载入一个字符串，然后调用strings\_not\_equal（）函数，顾名思义比较是否相等，返回值在寄存器eax中，判断eax中的值是否为0，如果不为0，则说明不相等，跳转到explode\_bomb；反之相等，退栈返回。

故而，我们要输入的字符串就是要与载入的字符串相等，于是我们通过x/s $rsi，查看那个字符串，得到结果Wow! Brazil is big.

Phase\_2: 首先是进栈，再将栈顶指针减0x28，接着就是栈的健壮等的操作，调用read\_six\_numbers（）函数，顾名思义读入六个数，然后比较第一个数是否为1，不相等，跳转到explode\_bomb。然后进入循环，将第一个数乘2与第二个数比较，不相等，跳转到explode\_bomb。若相等，则比较是否超过范围，即这六个数。若没超，则继续循环比较之后的几个数，直到比较完这六个数。进行退栈返回。

故而，第一个数必须是1，剩下的是前一个数的二倍所以这六个数是1 2 4 8 16 32

Phase\_3:开头是常规的进栈和健壮工作。通过查看寄存器rsi，里面是“%d%d”，即要输入两个数，然后调用了输入函数，寄存器eax里存储的是输入数字的个数，eax中的数小于2则爆炸，确保输入至少两个数。第一个数要小于7，反之就跳转到explode\_bomb。因为是无符号比较，所以第一个数只能是0-7，根据跳转，或者通过GDB的单步调试si，我们可以发现0跳转到0x5555555556e1，即第二个数要等于0x2d6才可以，同理，1对应于0x4e,2对应于0x330,3对应于0x1a1,4对应于0x2dc,5对应于0xf3,6对应于0xa1,7对应于0x62.这就是一个switch-case语句。

故而，可以输入0 726 或者1 78 或者2 816 或者3 417 或者4 732 或者5 243 或者6 161 或者7 98

phase\_4：开头是常规的进栈和健壮工作。通过查看寄存器rsi，里面是“%d%d”，即要输入两个数，然后调用了输入函数，寄存器eax里存储的是输入数字的个数，判断eax中的数是否等于2，不等则跳转爆炸。确保输入两个数。同时第一个数要小于等于14。调用func4（）函数。最后判断eax和rcx指向的数是否为5。是则结束。我们可以发现第一个数存在edi中，第二个数存在rcx中。

故而，第二个数就是5。从结果入手为了使eax中为5，则如果ecx中的数大于第一个数，则最后eax为2\*eax，必为偶数，不符。如果ecx中的数等于第一个数，则最后eax为0，显然不对。所以第一次必然是ecx小于第一个数，ecx中的数第一次比较的时候是7，故第一个数大于7。esi中存了6，又进入func4函数。此时第二次比较的时候ecx为10，为了尽快结束递归，我选择此时就结束，那ecx就要等于第一个数，10>7，满足条件。所以10 5为一种答案。

phase\_5：开头是常规的进栈和健壮工作。通过查看寄存器rsi，里面是“%d%d”，即要输入两个数，然后调用了输入函数，寄存器eax里存储的是输入数字的个数，eax中的数小于2则爆炸，确保输入至少两个数。将第一个数与15取位与，且此时不能等于15。rsi是一个线性表的表头。进入循环，需要循环15次，并且最后一个数是15.

首先，弄清楚这个数组，我们可以发现，数组存的是10,2,14,7,8,12,15,11,0,4,1,12,2,9,6,5这16个数，为了使15为第15个数我们倒推，发现只有当从第五个数开始的时候才可以，所以第一个数为5。第二个数是这15次循环的15个数的和，显然为115。所以输入5 115

phase\_6：开头是常规的进栈和健壮工作。调用read\_six\_numbers（）函数，顾名思义读入六个数，然后进入循环，循环六次内层循环是保证六个数不相等。六个数都小于等于六，结束循环。进入下一个循环，循环六次。将六个结构按照输入的六个数的大小，按照升序的顺序存入栈中，结束循环。然后将结构连接起来，形成链表。最后循环比较结构中的数是非递减序。

故而，为了按照最后结构中成员数的非递减序，我们查看结构中的数值分别是430,850,221,792,497,292，所以只有按照3 6 1 5 4 2 顺序才可以。