第一题:

1. 首先在linux下使用objdump生成bomb的反汇编文件。

2. 使用vim查看该文件。

3. 使用查找定位到main的汇编代码，之后按照源c程序查找phase\_1的位置。

4. 在转进phase\_1的子程序前可以看出，有一些控制输入的代码，并且会将输入字符串的首地址保存在eax中。

5. phase\_1子程序中调用了string\_equals子程序进行比较，并将比较的结果保存在eax中，因此只需要传入完全相同的字符串，确保输出eax为1（数据串相同）。

6. 在执行string\_equals子程序时将0x804a204的数据段压栈，因此可以推测参考字符串存储在该地址处。

7. 使用gdb查看该位置处的数据，并将相应的数据手动转换成Ascii码即可。

8. 根据转换可以看出参考字符串为：And they have no disregard for human life.

第二题:

1. 同解决第一题的步骤1-4，找到对应的phase\_2代码段所在的位置。

2. 首先可以看出，在本子程序中调用了read\_six\_numbers的子程序，因此，推测程序的前半段是将输入的字符串（首址保存在eax中）解析为6个数字。经过gdb调试可以验证猜测正确，将字符串解析为6个数字，具体的方法就是使用文件重定向，将输入的字符串通过“%d %d %d %d %d %d”再次重新读取（具体参考read\_six\_numbers的反汇编代码）。

3. 在读取6个数字之后，可以看出代码比较第一个数字是否为0，第二个数字是否为1，并且判断第三个数字是否为第一个数字0和第二个数字1的和，之后是一个循环过程判断循环次数是否为4，因此猜测该验证结果可能为n=6的斐波拉且数列：0 1 1 2 3 5

4. 经过验证可以证明猜测正确。

5. 本题的破解主要按照防止调用爆炸子程序的线索进行，可以采用逆推的方式，找到程序安全退出的路线，这个方法在之后的题目中也经常用到。

第三题:

1. 同解决第一题的步骤1-4，找到对应的phase\_3代码段所在的位置。

2. 在程序的开始看到了与read\_six\_numbers中读入数字类似的代码段，可以推测程序一开始也是将字符串解析为几个数字，根据两次的压栈eax可以猜测为数字为两个。使用两个数字测试验证正确。

3. 调用sscanf后，eax保存着读取的个数，根据代码可知需要个数大于1，否则bomb。

4. 根据任务书的提示，可以明显地找到代码中地switch提示部分，并且使用gdb查看0x804a080附近的内存信息，可以发现是一个明显的switch转移表。

5. 根据逆推找到程序正确退出的唯一路径，可以发现，只有当case为1时才能正常退出，并且根据case为1的代码操作可以看出是在比较第二的数值是否为程序经过计算得到的数值，因此根据计算可以得出，参考数值为-1233。

6. 综上数据为 1 -1233。成功。

第四题:

1. 同解决第一题的步骤1-4，找到对应的phase\_4代码段所在的位置。

2. 在子程序开始时看到了与phase\_3完全相同的代码段，因此可以推测，也是将字符串转换为两个数字。从之后比较eax是否为2也可以看出猜测正确。

3. 根据逆推可以看出如果想要正常退出，需要执行完成func4后eax（返回值）为1，并且第二个输入的数字为1。

4. 可以进入func4查看具体的递归过程，但是递归过程较为复杂，不建议直接按照代码推理，根据提示可以看出，输入的第一个数字要小于等于0xe，因此可以使用暴力求解的方式进行测试（猜数）。经过猜测验证第一个数字为11（0xb）。

5. 本题目的递归算法及其复杂，不过幸亏输入的数字只有一个而且数字比较小，因此可以成功破解。

第五题:

1. 同解决第一题的步骤1-4，找到对应的phase\_5代码段所在的位置。

2. 在进入程序时可以看出调用string\_length的子程序获取输入的字串的长度，根据之后的比较可以看出串的长度需要为6。

3. 之后是一个6次循环，每一次循环中都会将首址为0x804a0a0的数组中的一个数据相加到ecx中，最后在退出时需要ecx等于0x3d。

4. 使用gdb查看该数组中每个的元素，可以看到：2 a 6 1 c 10 9 3 4 7 e 5 b 8 f d（均为16进制数据）。因此在这里选取 a c 10 4 b 8。（也可以选择其它的，只需要保证和为0x3d）对应的位置为1、4、5、8、c、d，因此使用相对应的ASCII码，只需要使用后四位即可，在这里选用“1458<=”（也可以使用其他的，只需要保证后四位即可）。

第六题:

1. 同解决第一题的步骤1-4，找到对应的phase\_6代码段所在的位置。

2. 程序开始的部分为读取6个数字，同第二题。设读取的6个数字分别为a b c d e f。

3. 通过后面的汇编代码分析可知，需要输入的6个数字在1-6之间选取，并且每一个数字与其它数字都不相同。

4. 之后是一个6次的循环，将a b c d e f分别替换为7-a 7-b 7-c 7-d 7-e 7-f。

5. 继续执行可以看出，代码将0x804c13c为首址的一片数组按照顺序读取内容（地址），并且排列的顺序与7-a 7-b 7-c 7-d 7-e 7-f相关。经过仔细的推导可以看出，7-a选取第7-a个地址放在一号位置，7-b选取第7-b个地址放在二号位置，以此类推，形成一定的排列，放在堆栈内。

6. 根据上一步形成的排列顺序形成相对应的链表内容。

7. 在之后的计算中通过逆推可以看出，链表的每两个相邻元素需要第一个元素的数据大于第二个元素的数据，否则bomb。因此可以看出需要指定a b c d e f的数字，以使排列顺序从大到小。

8. 使用gdb查看0x804c13c处的数据，可以看到：①4a ②19d ③12a ④197 ⑤129 ⑥ba（数据均为16进制数据）。排列可得从大到小的顺序：②④③⑤⑥①，因此7-a=2 7-b=4 7-c=3 7-d=5 7-e=6 7-f=1，即a=5,b=3,c=4,d=2,e=1,f=6。验证成功。