Jessica Fang
Phase 1 can be defused by running until *0x400ee4. At address line 400ee4, enter x/s 0x402400 to show the string that is stored in the address which our input will be compared to.

Phase 2 can be done in the same way. From address line 400f05, it is clear that the input will be 6 numbers. Run until *0x400f1c. At address line 400f1c, enter i r to see all the registers and check the value being stored in %eax (listed as %rax). Then run until *0x400f1c again for the second number. Do this another 4 times to get all 6 numbers, entering the "correct" input if the bomb explodes or if you restart because you know the bomb will explode because you entered "incorrect" input. Alternatively, after you notice that 4001fc is the compare, you can figure out that the next number is double the previous number by looking at 400f1a.

<phase_1>:

400ee0	sub	$0x8,%rsp	push 0x8 bytes from the stack (%rsp)
400ee4	mov	$0x402400,%esi	move 0x402400 in %esi
400ee9	callq	401338 <strings_not_equal>	call function strings_not_equal at address 401338
400eee	test	%eax,%eax	sets zero flag to 1 if the AND operation is 0 (FALSE)
400ef0	je	400ef7 <phase_1+0x17>	jump to address 400ef7 if zero flag is 1
400ef2	callq	40143a <explode_bomb>	call the function explode_bomb at address 40143a
400ef7	add	$0x8,%rsp	pop 0x8 bytes from the stack
400efb	retq		return the value at the top of the stack

<phase_2>:

400efc	push	%rbp	store %rbp in stack
400efd	push	%rbx	store %rbx in stack
400efe	sub	$0x28,%rsp	push 0x28 bytes from the stack
400f02	mov	%rsp,%rsi	move the contents of rsp to rsi
400f05	callq	40145c <read_six_numbers>	call the function read_six_numbers at address 40145c
400f0a	cmpl	$0x1, (%rsp)	compare 4 bytes of 0x1 to the value stored in %rsp
400f0e	je	400f30 <phase_2+0x34>	jump to address 400f30 if zero flag is 1
400f10	callq	40143a <explode_bomb>	call the function explode_bomb at address 40143a
400f15	jmp	400f30 <phase_2+0x34>	jump to address 400f30
400f17	mov	-0x4(%rbx),%eax	move the contents of %rbx minus 0x4 to %eax

400f1a	add	%eax,%eax	double the value of %eax (add %eax to %eax and store into %eax)
400f1c	cmp	%eax,(%rbx)	compare the contents of %eax to %rbx
400f1e	je	400f25 <phase_2+0x29>	jump to address 400f25 if zero flag is 1
400f20	callq	40143a <explode_bomb>	call the function explode_bomb at address 40143a
400f25	add	$0x4,%rbx	add 0x4 to the content of rbx
400f29	cmp	%rbp,%rbx	compare the contents of %rbp to %rbx
400f2c	jne	400f17 <phase_2+0x1b>	jump to address 400f17 if zero flag is 0
400f2e	jmp	400f3c <phase_2+0x40>	jump to address 400f3c
400f30	lea	0x4(%rsp),%rbx	change the address of %rbx to that of %rsp plus 0x4
400f35	lea	0x18(%rsp),%rbp	change the address of %rbp to that of %rsp plus 0x18
400f3a	jmp	400f17 <phase_2+0x1b>	jump to address 400f17
400f3c	add	$0x28,%rsp	pop 0x28 bytes from the stack
400f40	pop	%rbx	remove %rbx from the stack
400f41	pop	%rbp	remove %rbp from the stack
400f42	retq		return the value at the top of the stack