Overview:
Charlotte Yamamoto (cyamam01), Logan Yuan (lyuan04)
We had bomb32
We did not have help from anyone else.
We spent approximately 12 hours on this assignment.

Defuse:
We get signal
1 2 6 24 120 720
0 i 361
17
mpmeka
5 6 3 2 1 4

Description:

1. Phase 1 reads a string and explodes if your input is not equal to the string stored at address 0x401e3c
2. Phase 2 reads in six numbers and explodes if the six numbers are not the first 6 numbers of the standard factorial sequence (starting at 1).
3. Phase 3 reads in a number, a character, and another number. Based on the first number read in, the function will jump to a specific statement in a switch statement and check if the other input matches some specific values in memory. It will explode if they do not match.
4. Phase 4 reads in a number *n* and explodes if the *n*th term of the Fibonacci sequence is not 2584 (the 17th term of the sequence when the 0th and 1st terms are 1).

Code:

```
/*****
Charlotte Yamamoto (cyamam01)
Logan Yuan (lyuan04)

Assignment: Bomb

This file contains the functions that we think are equivalent to the machine
code for phases 5 and 6 (alongside function contracts for the helper
functions).

*****/

#include <stdbool.h>

extern void explode_bomb(); /* explodes the bomb */
extern int string_length(char* input); /* returns length of string */
extern bool strings_not_equal(char* s1, char* s2); /* self-explanatory */
extern void read_six_numbers(char* input, int* first); /* self-explanatory */

/* table used for phase_5 used for encryption*/
char* table = "isrveawhobpnutfg";

/*
our guess as to what the fields inside the node struct are for phase 6
*/
struct node
{
        short num;
        int magic_number;
        struct node* next;
};

/* first node in linked list in memory */
extern struct node node1;

/********** phase_5 ********
 *
 * checks if the input is equal to "titans" after decryption
 *
 * Parameters:
 *      char* : string representing what user put in
 *
 * Return: Void
 *
 * Expects
 *      Input is not null
```

```
 * Notes:
 *      it will explode if length of input is not 6, and if input is not equal
 *      to titans after decryption
 ************************/
void phase_5(char* input)
{
        if(string_length(input) != 6){
                explode_bomb();
        }
        for(int i = 0; i < 6; i++){
                int last_four_bits = input[i] & 0xf; /* mask each letter */
                input[i] = table[last_four_bits]; /* lookup in table */
        }
        if(strings_not_equal(input, "titans")){
                explode_bomb();
        }
        return;
}

/********** phase_6 ********
 *
 * reads in six numbers in a mysterious order specified by the magic numbers.
 * explodes if the numbers are in the wrong order, or if there are any repeat
 * numbers, or if any number is > 6
 *
 * Parameters:
 *      char* : string representing what user put in
 *
 * Return: Void
 *
 * Expects
 *      Input is not null
 * Notes:
 *      it will explode if less than 6 numbers are read
 *      it will explode if any number is greater than 6
 *      it will explode if any number is repeated
 ************************/
void phase_6(char* input)
{
        int arr[6];
        read_six_numbers(input, &arr[0]);

        for(int i = 0; i <= 5; i++){
                int curr = arr[i];
                if(curr - 1 > 5){ /* check if each number - 1 > 5*/
                        explode_bomb();
                }
```

```
                for(int j = i + 1; j <= 5; j++){ /* check for duplicates */
                        if(curr == arr[j]){
                                explode_bomb();
                        }
                }
        }

        struct node* nodes[6];

        /*
         find nodes in memory corresponding to the input numbers, stores
         addresses in nodes[] by the order they were inputted
        */
        for(int i = 0; i != 6; i++){
                int curr = arr[i];
                struct node* curr_node = &node1;
                for(int j = 1; j < curr; j++){
                        curr_node = curr_node->next;
                }
                nodes[i] = curr_node;
        }

        /* align the nodes */
        nodes[0]->next = nodes[1];
        nodes[1]->next = nodes[2];
        nodes[2]->next = nodes[3];
        nodes[3]->next = nodes[4];
        nodes[4]->next = nodes[5];
        nodes[5]->next = NULL;

        struct node *curr_node = nodes[0];

        /*
         ensure that magic number of each node is greater than the the
         magic number of the node it points to
        */
        for (int i = 5; i > 0; i--) {
                int curr = curr_node->magic_number;
                curr_node = curr_node->next;
                int next = curr_node->magic_number;
                if (curr < next) {
                        explode_bomb();
                }
        }
}
```