Overview

Name: Armaan Sikka & Jeremy Lawrence
UTLN: asikka01 & jlawre09

Bomb Number: 68

Help and Collaboration: We received TA help because we were having trouble doing “run” on bomb with gdb. We solved the issue with “chmod 700 bomb” which made an executable that we were able to “run” bomb with.

Hours Spent: 30 hours

Defuse

Main screen turn on
1 2 3 1 2 3
1 550
9 Zero_Wing
cbbbbd
167
35

Description

Phase 1: This phase reads in a string and calls explode_bomb() if the input string doesn’t match a pre-set string.

Phase 2: This phase reads in 6 long integers and calls explode_bomb() if six longs are not read, the first and fourth numbers aren’t equal, the second and fifth numbers aren’t equal, or the third and sixth numbers aren’t equal.

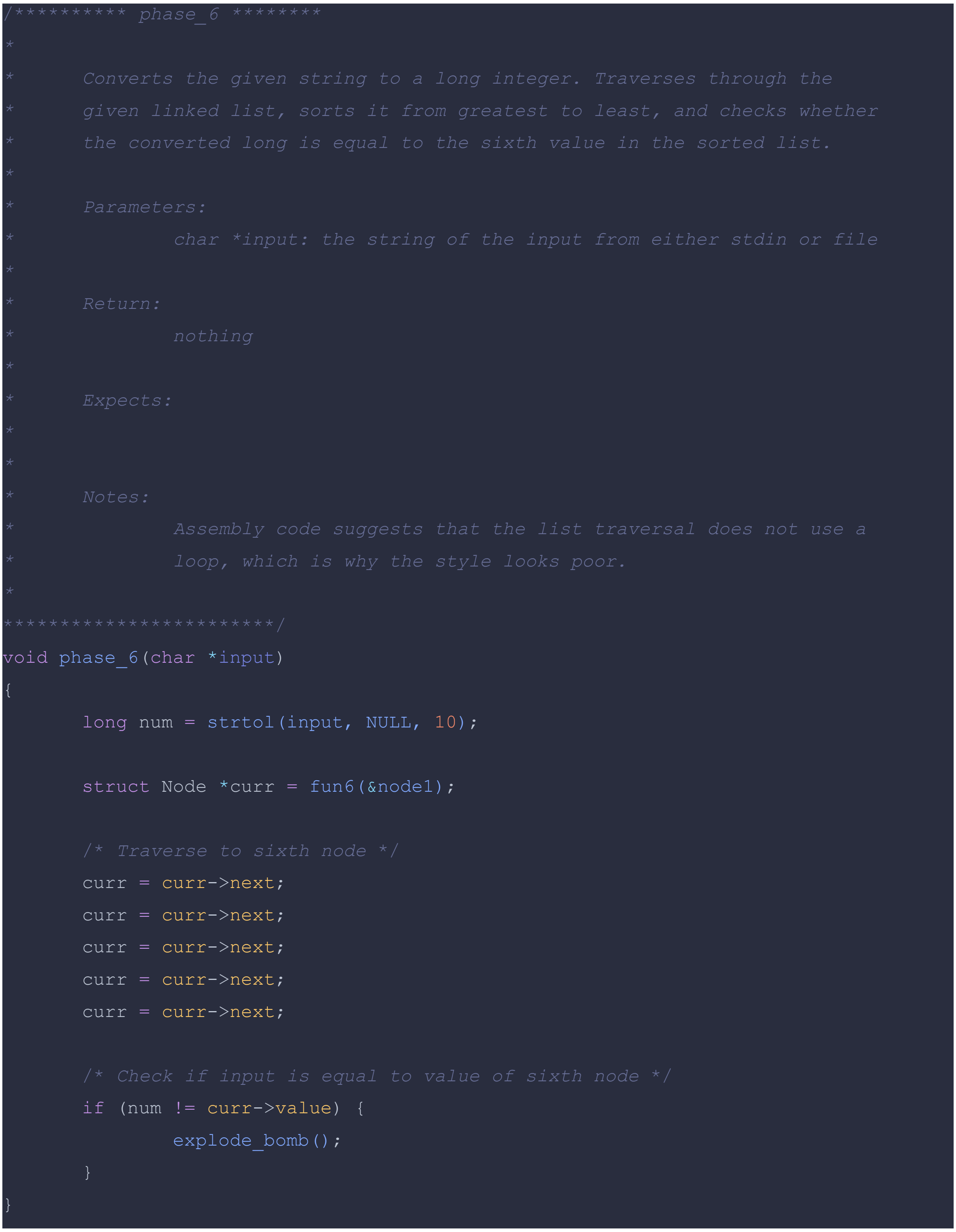
Phase 3: This phase reads in 2 long integers, the first of which must be between 1 and 7, inclusive. This number is used to determine a case in a switch-case operation and if the second number doesn’t match the predetermined matching number for that case, the function calls explode_bomb().
Possible Inputs: “7 91”, “6 460”, “5 964”, “4 394”, “3 840”, “2 998”, “1 550”

Phase 4: This phase reads in a number n, which must be positive or else explode_bomb() is called. A recursive function, func4, is called which computes n!, and if func4 does not return 9!, explode_bomb() is called.

Code

```
/*
*       phase5.c
*       by: Armaan Sikka & Jeremy Lawrence
*       utln: asikka01 & jlawre09
*       date: 11/3/24
*       assignment: bomb
*
*       summary:
*               Includes the code for phases 5 and 6 and the helper function
*               for phase_6, func_6.
*
*/

#include <stdlib.h>

/* extern declarations of functions not necessary to code */
extern void explode_bomb();
extern int string_length(char *s);

/* Hash table required for phase_5 */
static int hash[16] = {2, 10, 6, 12, 1, 16, 9, 3, 4, 7, 7, 5, 11, 8, 15, 13};

/* The structure of each Node for phase_6 */
struct Node {
       long value;
       struct Node *next;
};

/* Initialise Nodes and set up Linked List */
struct Node node9 = { .value = 121, .next = NULL };
struct Node node8 = { .value = 473, .next = &node9 };
struct Node node7 = { .value = 165, .next = &node8 };
struct Node node6 = { .value = 167, .next = &node7 };
struct Node node5 = { .value = 553, .next = &node6 };
struct Node node4 = { .value = 527, .next = &node5 };
struct Node node3 = { .value = 549, .next = &node4 };
struct Node node2 = { .value = 941, .next = &node3 };
struct Node node1 = { .value = 146, .next = &node2 };

/********** phase_5 ********
*
*       Takes in a string, and the bomb explodes if it is not 6 characters
*       long. If it is, each character is hashed and the sum of their
*       associated values is counted. If the cumulative value is 37, the phase
*       is defused. Otherwise, the bomb explodes.
*
*       Parameters:
*               char *input: the string of the input from either stdin or file
*
*       Return:
*               nothing
*
*       Expects:
*               A string of length 6, where the sum of the hashed values of
*               each character is 37.
*
*       Notes:
*
*
************************/
void phase_5(char *input)
{
        if (string_length(input) != 6) {
                explode_bomb();

        } else {
                int total = 0;

                for (int i = 0; i < 6; i++) {
                        char c = *input;
                        total += hash[c % 16];
                        input++;
                }

                if (total != 37) {
                        explode_bomb();
                }
        }
}

/********** fun6 ********
*
*       Sorts the given linked list from greatest to least.
*
*       Parameters:
*               struct Node *head: pointer to the head of the list before sort
*
*       Return:
*               struct Node *: a pointer to the new head with list in order
```

```c
*
*       Expects:
*
*
*       Notes:
*
*
************************/
struct Node *fun6(struct Node *head)
{
        /* Head of the new sorted list */
        struct Node *newHead = NULL;

        struct Node *curr = NULL;
        struct Node *temp = NULL;

        /* Traverse the unsorted list */
        while (head != NULL) {
                /* Detach head of unsorted list */
                curr = head;
                head = head->next;
                curr->next = NULL;

                /* Curr node becomes new head if it is the max value so far */
                if (newHead == NULL || curr->value > newHead->value) {
                        curr->next = newHead;
                        newHead = curr;

                } else { /* Curr node is not the max value so far */

                        temp = newHead;

                        /* Insert curr node into new sorted list */
                        while (temp->next != NULL &&
                                temp->next->value > curr->value) {
                                 temp = temp->next;
                        }

                        curr->next = temp->next;
                        temp->next = curr;
                }
        }
        return newHead;
}
```



```c
/********** phase_6 ********
*
*       Converts the given string to a long integer. Traverses through the
*       given linked list, sorts it from greatest to least, and checks whether
*       the converted long is equal to the sixth value in the sorted list.
*
*       Parameters:
*               char *input: the string of the input from either stdin or file
*
*       Return:
*               nothing
*
*       Expects:
*
*
*       Notes:
*               Assembly code suggests that the list traversal does not use a
*               loop, which is why the style looks poor.
*
************************/
void phase_6(char *input)
{
        long num = strtol(input, NULL, 10);

        struct Node *curr = fun6(&node1);

        /* Traverse to sixth node */
        curr = curr->next;
        curr = curr->next;
        curr = curr->next;
        curr = curr->next;
        curr = curr->next;

        /* Check if input is equal to value of sixth node */
        if (num != curr->value) {
                explode_bomb();
        }
}
```