Analysis of Bomb

Overview:

Jack Adkins (jadkin05), Nate Pfeffer (npfeff01)
11/5/2024

We had bomb73

We used www.tutorialspoint.com to understand the functions in the c standard library.
We also used the AMD instruction set manual and Google searches to learn what certain assembly instructions did.

We spent approximately 15 hours on the assignment.

Defuse lines:

What you say
1 2 6 24 120 720
5 -302
6 Zero_Wing
7 93
302
22

Phase 1-4 Descriptions:

Phase 1:

- Phase 1 reads a string and explodes the bomb if the string is not "What you say"

Phase 2:

- Phase 2 needs 6 numbers as input to avoid exploding the bomb. The first number has to be one, and it starts a multiplier at 1. It then starts a loop where it increments teh multiplier by one, then multiplies the previous number (starting with 1) by the multiplier and compares it to the next number from input. If those numbers are not the same then the bomb will explode. The loop stops after the multiplier multiplies by 6. Phase is passed if the loop ends without exploding the bomb.

Phase 3:

- Phase 3 needs 2 numbers as input. It has a series of arithmetic steps, and the size of the first number determines how far along in those arithmetic steps you start at. When the arithmetic is complete, the final value calculated needs to be equal to the second number provided as input.

Phase 4:

- Phase 4 takes a single number as input, if the number is less than or equal to 0 then the bomb explodes. A recursive factorial function is called with the provided number as input. The value returned from the factorial function needs to equal 720.

Phases 5-6 C Code:

```
#include <stdio.h>
#include <stdlib.h>

extern void explode_bomb();

void phase_5(char *input)
{
  int array[16] = {10,2,14,7,8,12,15,11,0,4,1,13,3,9,6,5};
  int index;
  int target;
  int numInputs = sscanf(input,"%d %d", &index, &target);
  if (numInputs <= 1) {
    explode_bomb();
  }
  if (index % 16 == 15) {
    explode_bomb();
  }
  int counter = 0;
  int running_sum = 0;
  while (index != 15) {
    counter++;
    index = array[index];
    running_sum += index;
  }
  if (counter != 12 || running_sum != target) {
    explode_bomb();
  }
}
```

```
typedef struct Node {
 int data;
 int node_number;
 struct Node *next;
} Node;

Node *fun6(Node *starter)
{
 struct Node *sorted = NULL;
 struct Node *current = starter;
 while (current != NULL){
  struct Node *next_node = current->next;
  struct Node *prev = NULL;
  struct Node *temp = sorted;
  while (temp != NULL && temp->data > current->data) {
   prev = temp;
   temp = temp->next;
  }
  if (prev == NULL) {
   current->next = sorted;
   sorted = current;
  } else {
   prev->next = current;
   current->next = temp;
  }
  current = next_node;
 }
 return sorted;
}

void phase_6(char *input)
{
 Node node9 = {193, 9, NULL};
 Node node8 = {369, 8, &node9};
 Node node7 = {302, 7, &node8};
 Node node6 = {651, 6, &node7};
 Node node5 = {568, 5, &node6};
```

```
  Node node4 = {261, 4, &node5};
  Node node3 = {540, 3, &node4};
  Node node2 = {315, 2, &node3};
  Node node1 = {871, 1, &node2};

  char *dummy;
  int entered = strtol(input, &dummy, 10);
  Node *final_node = fun6(&node1);
  for (int i = 0; i < 6; i++) {
    final_node = final_node->next;
  }
  if (entered != final_node->data){
    explode_bomb();
  }
}
```