Testing und Debugging - Übungsblatt 10:

Gruppe 11:
Manuel Neubauer - 79358
Hueseyin Selcuk - 79375
Abdullah Yildiz - 79669
Daniel Zahl - 79356

1. **Memory Probleme**

    1. Bufferoverflow (50 Punkte)

        1. Welches Fehlverhalten können Sie an der Ausgabe der center Funktion in der Datei ergebnis.txt neben dem Auslösen der usage Funktion noch beobachten? (30 Punkte):

            Erwartet:
            Bei Eingabe des Strings "Hallo" sollte der String zentriert ausgegeben werden.

            Beobachtet:
            Keine Ausgabe.

            Fehlerursache:
            Der Buffer wurde auf den Stackzugewiesen durch den folgenden Befehl:
            char buf[81];

            Dies führt dazu, dass nach einem "return" der Funktion die Adresse überschrieben wird. Deshalb ist das Verhalten der Ausgabe undefiniert. Anstatt den Buffer auf den Stack zuzuweisen, soll man durch den folgenden Befehl:
            **char buf = malloc(81 * sizeof(char));*
            die Daten auf den Heap zuweisen. Dadurch wird die Adresse nicht überschrieben, und man erhält die gewünschte Ausgabe.

        2. Welche Zeile in der center.c Datei verursacht den Bufferoverflow? Begründen Sie Ihre Antwort. Durch welche Änderung in dieser Zeile kann man den Bufferoverlow verhindern? (10 + 10 Punkte):

            Der Bufferoverflow entsteht durch folgende Zeile:
            strcpy(buf + i, s);

            Die Funktion strcpy prüft nicht die zu kopierende Länge der Strings der Quelle, und die Länge des Zieles. Ist nun die Länge der Quelle größer als der des Zieles, entsteht ein Bufferoverflow und es werden Bereiche überschrieben (Instructionpointer & return) die nicht überschrieben werden dürfen.

            Durch Benutzen der Funktion "strncpy" kann dieses Problem behoben werden:
            strncpy(buf + i, s, 80-i);
            buf[81] = '\0';

Da strncpy beim abschneiden des Strings nicht automatisch einen Stringterminator "\0" setzt, muss dieser noch hinzugefügt werden.

2. Lokalisieren der Variablen im Speicher (100 Punkte)

1. In welchem Teil des Speichers sind die Variablen length, i und buf der Funktion center angesiedelt, wenn diese Funktion ausgeführt wird? (10 Punkt)

   Die Variablen sind im Stack enthalten.

2. In welchem Zeitraum während der Ausführung des Programms existieren die o. g. Variablen? (10 Punkte)

   Solange die Funktion ausgeführt wird.

3. Wohin zeigt die Variable msg nachdem die Zeile 77 von uebung10.c ausgeführt wurde? Worauf nach Zeile 78 von uebung10.c? Was steht jeweils an diesen Speicherstellen? (5 + 5 + 5 Punkte)

   Nach der Zeile 77 von uebung10:
   $2 = 0x555555757670 ' ' <repeats 37 times>, "hallo"

   Nach der Zeile 78 von uebung10:
   $2 = 0x555555757670 ' ' <repeats 37 times>, "hallo", '\000' <repeats 38 times>

4. Welche Werte kann i annehmen, bevor die Variable in der Zeile mit strcpy benutzt wird ? (10 Punkte)

   i kann die Zahlen von 0 bis 40 annehmen.

   Für einen leeren String: "" (length = 0)
   Dadurch haben wir (78-0)/2,was 39 entspricht. Der Loop läuft bis i <= 39. Also wird er im letzten Schritt auf i = 40 erhöht und beim check, ob i<=40 ist bricht der loop dann ab. Dies ist die Obergrenze für i mit i = 40.

   Für einen String >= 80 (length >= 80)
   Dadurch haben wir (78-80)/2, was –1 ergibt. Da i = 0 wird beim prüfen der Bedingung 0 <= -1 der Loop verlassen. Dadurch ist i = 0 der kleinste Wert, welchen i annehmen kann.

   Bsp:
   Für den String "hallo":
   length = 6
   78 – length = 72
   72 / 2 = 36
   i <= 36

   Im Beispiel könnte i die Zahlen 0-36 (insgesamt 37 Zahlen) annehmen.

5. In welchem Teil des Speichers sind die Variablen f, f0 und f1 der Funktion fib angesiedelt, wenn diese Funktion ausgeführt wird? (10 Punkt)

   Die Variablen sind im Stack enthalten.

6. In welchem Zeitraum während der Ausführung des Programms existieren die o. g. Variablen? (10 Punkte)

   Solange die Funktion ausgeführt wird.

7. Auf welche Adresse zeigt die Variable f aus uebung10.c vor und nach der Ausführung der fib Funktion? Auf welche Adresse zeigt die Variable f in der fib Funktion vor dem return? (in gdb jeweils "p &f" an geeigneter Stelle ohne Anführungszeichen ausführen) (5 + 5 + 5 Punkte)

   Vor der Ausführung:

   1: f = 1431653200
   2: &f = (int *) 0x7fffffffe400

   Nach der Ausführung:

   1: f = 2
   2: &f = (int *) 0x7fffffffe400

8. Welche Werte kann f in fib annehmen, bevor sie mit return zurückgegeben wird? (10 Punkte)

   Falls die Funktion richtig aufgerufen wird (mit einer kleinsten, mittleren und größten Zahl) wird die Fibonacci Funktion die Zahlen für mittlere Zahl - 1 berechnet.
   Ist die Zahl jedoch nicht größer 1, kann f eine beliebige postitive oder negative Zahl annehmen. Der Grund dafür ist, f wurde bisher nur deklariert (int f) und kein Wert zugewiesen. Da die while Schleife erst bei n>1 f einen Wert zuweist, und sonst das Programm mit einem return beendet, kann kein Wert zugeordnert werden.

9. Ruft man das Programm mit negativen Zahlen für die ersten drei Parameter auf, so gibt fib unerwartete Werte zurück. Warum ist das so? Wo liegt der Fehler in fib.c? (10 Punkte)

   Bei einem negativen m wird die Bedingung der while Schleife übersprungen und das uninitialisierte f ausgegeben.