תרגיל בית 2 חלק יבש – מבוא לתכנות מערכות

תאריך הגשה: 17.5.17

מגישים:

סהר כהן, ת.ז: 206824088
עודד ורמשטיין, ת.ז: 209320332

שאלה 1:

1) ***assert(argc == 2)***: השימוש תקין.

הסבר: מעצם השימוש ב assert, המתכנת מבצע הנחה כי התוכנית שלו תקרא בתוספת פרמטר יחיד. אחרי הוספת הדגל DNDEBUG: גם אם מתכנת אחר המפעיל את התוכנית יקרא לה בתוספת ארגומנטים נוספים – לא יתבצע בהם שימוש כלל, ולכן לא תהיה השפעה.

2) ***assert(s! = NULL)***: השימוש לא תקין.

הסבר: אחרי הוספת הדגל DNDEBUG: תיפתח פרצת באגים בקוד כמקרה של קריסת התוכנית מעצם חריגת זיכרון. על פניו, אין בעיה בשימוש ב – assert, אך בפני עצמו אין הוא משמש תחליף לבדיקת ערך ההחזרה null במקרה של בעיית זיכרון.

3) ***assert(strlen(s) < N)***: השימוש לא תקין.

הסבר: assert נועד לשימוש המתכנת בלבד, ואין הוא משמש אינדקציה למשתמש חיצוני באשר להנחות הקוד שלנו. יש להדפיס הודעת שגיאה מתאימה למשתמש אשר מודיעה לו על הגבלות גודל הקלט הדרושות בקוד.

4) ***assert(! * (s + strlen(s)))***: השימוש תקין.

הסבר: המתכנת בודק שתי הנחות בזמן תכנון הקוד על מנת שיוכל להסתמך על נכונותן בהמשך כתיבת הקוד. במקרה זה, מדובר בהנחה לגיטימית: קיום ה null terminator בסוף המחרוזת, כתוצאה מפורמט הקליטה s%. משום שניתן לצפות שהנחה זו תתקיים – היא גם לא תקריס את התוכנית.

5) ***assert(atol(s))***: השימוש לא תקין.

הסבר: פרצת באגים רצינית וקשה לאיתור: atol תחזיר 0 אם מחרוזת הקלט אינה תווים מספריים, וגם אם הקלט הוא מחרוזת המורכבת מ - "0" בלבד. במידה ואחד מן השניים מתקיים, תתבצע חלוקה באפס בשורה הבאה חרף העבודה ש assert הצליח והתוכנית תקרוס. גם במידה ו assert מקריס את התוכנית: אחריות המתכנת להודיע למשתמש על הנחת הקוד שלו (שימוש במחרוזת המורכבת ממספרים בלבד), בדומה לסעיף 3.

6) ***assert(s != NULL)***: השימוש לא תקין.

הסבר: אחרי הוספת הדגל DNDEBUG: תיפתח פרצת באגים בקוד במקרה בו הפונקציה מקבלת מצביע שערכו null ומנסה לעשות לו dereferencing: התוכנית קורסת. יש לטפל במקרה זה באמצעות שגיאה ייעודית, ובמידה וקיים טיפול בשגיאה זו: השימוש ב assert חוקי, משום שאין הוא מחליף את הבדיקה ההכרחית.

שאלה 2:

קוד התוכנית הדרוש:

hw2_dry_q2.c

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

/* Gets an array of strings and its length. Returns a string which is the
 * concatenation of all the strings in the array, in their original order.
 * Will return NULL if there isn't enough sufficient memory.
 * Space complexity: O(n x m), where 'n' is the length of the original array,
 * and 'm' is the length of the longest string in the array */
char *flat_text(char **words, int n) {
    if(!words) return NULL; //Error: Illegal strings array
    /*Variables:
    total_length: counts the total amount of characters in the complete, connected array.
    connected_word: stores the concatenation of the first i strings in the array.
    word_copy: stores the concatenation of the first i-1 strings in the array in a separate copy */
    int total_length = strlen(words[0]);
    char *connected_word = malloc(++total_length);
    if(!connected_word) return NULL; //Error: insufficient memory
    strcpy(connected_word, words[0]);
    char *word_copy = NULL;
    for(int i=1; i<n; i++) {
        /* In each iteration of this loop, the connected_word string concatenates the next string in the array
         * to the previous ones. word_copy copies the value of the result string, and then the result string is
         * freed to allow allocation for bigger space (enough to store the next string) */
        if(!words[i]) continue; //Current string is empty: move to the next string in the array.
        word_copy = malloc(++total_length);
        if(!word_copy) return NULL; //Error: insufficient memory
        strcpy(word_copy, connected_word);
        total_length+=strlen(words[i]);
        free(connected_word); //connected_word is now pointing at NULL
        connected_word = malloc(++total_length);
            if(!connected_word) return NULL; //Error: insufficient memory
        strcpy(connected_word, word_copy);
        strcpy(connected_word+strlen(word_copy), words[i]);
        free(word_copy); //word_copy is now pointing at NULL
    }
    return connected_word;
}

int main() {
    char *words[4] = {"Hello", "To", "234122", "Matam"};
    char *p = flat_text(words, 4);
    if (p!=NULL)
    {
        printf("\n%s\n", p);
        free(p);
    }
    return 0;
}
```

דוגמאות הרצה:

```
HelloTo234122Matam

Process returned 0 (0x0)   execution time : 0.016 s
Press any key to continue.
```

```c
        strcpy(connected_word+strlen(word_copy), words[i]);
        free(word_copy); //allocation: freed!
    }
    return connected_word;
}

int main() {
    char *words[4] = {"Hello", "To", "234122", "Matam"};
    char *p = flat_text(words, 4);
    if (p!=NULL)
    {
        printf("\n%s\n", p);
        free(p);
    }
    return 0;
}
```

C:\CBproject\temp4\matam2_dry.exe

```
HelloToThePersonWhoChecksThisCode!

Process returned 0 (0x0)   execution time : 0.016 s
Press any key to continue.
```

```c
    }
    return connected_word;
}

int main() {
    char *words[8] = {"Hello", "To", "The", "Person", "Who", "Checks",
    "This", "Code!"};
    char *p = flat_text(words, 8);
    if (p!=NULL)
    {
        printf("\n%s\n", p);
        free(p);
    }
    return 0;
}
```

```
ThankYouForYourPatience:)

Process returned 0 (0x0)   execution time : 0.016 s
Press any key to continue.
```

```c
        strcpy(connected_word+strlen(word_copy), words[i]);
        free(word_copy); //allocation: freed!
    }
    return connected_word;
}

int main() {
    char *words[6] = {"Thank", "You", "For", "Your", "Patience", ":)"};
    char *p = flat_text(words, 6);
    if (p!=NULL)
    {
        printf("\n%s\n", p);
        free(p);
    }
    return 0;
}
```