שגיאות תכנות

1. השורה: `assert(!s);`

שורה זו תעצור את התוכנית אם s הוא אינו NULL ותמשיך אם s הוא כן NULL. לכן זוהי שגיאה זה הפוך מהרצוי כי רוצים להמשיך אם המחרוזת לא NULL.

2. השורה: `char* out = malloc(LEN*times);`

לא מוקצה מקום לתו '\0' בסוף המחרוזת out ולכן זוהי שגיאה.

3. השורה: `char* out = malloc(LEN*times);`

לא נבדק לאחריה האם הmalloc הצליח או שמא הוחזר NULL כי לא היה מספיק מקום בזיכרון. זה יכול לגרום לכך שבהמשך התוכנית ניגש למקומות בזיכרון שלא הוקצו עבורנו. יש לציין כי ישנו assert לבדיקת הצלחת הmalloc אך זה אינו מספיק, משום שבזמן הריצה הassert לא יבדוק זאת ולכן יש דרישה לבדיקה באמצעות if.

4. השורה: `for(int i = 0; i <= times; i++)`

הלולאה רצה times+1 פעמים במקום times פעמים. לפי תוכן הלולאה נסיק שתוכן s יועתק times+1 פעמים לout מה שיגרור חריגה מהזיכרון של out.

5. השורות:

```
out = out + LEN;
strcpy(out, s);
```

השורות לא בסדר הנכון כיוון שמה שיקרה בפועל שההעתקה תתחיל מהמקום out+LEN ולא מתחילת המחרוזת out.

6. השורה: `return out;`

מכיוון שבלולאה המצביע למחרוזת out מתקדם ולא מצביע זמני, בסוף הפעולה נחזיר את המצביע כשאר הגיע למקום כלשהו במחרוזת ונאבד חלק מהמחרוזת.

שגיאות קונבנציה

1. קונבנציית שמות מזהים ובפרט שמות משתנים מופרת בשורה:

```
int LEN = strlen(s);
```

הקונבנציה היא ששמות משתנים תמיד באותיות קטנות בלבד וכאן המשתנה בעל אותיות גדולות.

2. קונבנציית מבנה הקוד ובפרט הזחות מופרת בקוד:
הסוגריים המסולסלים אינם מוזחים כמו שצריך- הן בלולאה והן בבלוק הפונקציה כולה. נוסף על כך אין הזחות בתוך הלולאה עצמה.

גרסה מתוקנת לקוד:

```
#include <stdlib.h>
#include <string.h>
#include <assert.h>
#define ZERO_SPOT 1

char* stringDuplicator(char* strToDuplicate, int times)
{
    assert(strToDuplicate);
    assert(times > 0);
    int len = strlen(strToDuplicate);
    char* out = malloc(len*times*sizeof(*out) + ZERO_SPOT);
    assert(out);
    for (int i = 0; i < times; i++)
    {
        strcpy(out + (len * i), strToDuplicate);
    }
    return out;
}
```