שגיאות קונבנציה:

1. char* stringDuplicator(char* s, int times){
סוגריים מסולסלים עבור בלוק של פונקציה יופיעו בשורות נפרדות.

2. int LEN = strlen(s);
שמות של משתנים מקומיים יופיעו באותיות קטנות.

3. assert(out);
לאחר malloc יש לבדוק תקינות פוינטר באמצעות if ולא assert.

4.
```
for (int i=0; i<=times; i++){
out = out + LEN;
strcpy(out,s);
}
```
הבלוק של לולאת for לא קיבל הזחה.

שגיאות תכנות:

1. assert(!s);
assert יוצא מהתוכנית כאשר הביטוי בו הוא false,לכן הפקודה הנ"ל תצא מהתוכנית אם s תקין.
לפיכך יש לכתוב את השורה כ assert(s);

2. char* out = malloc(LEN*times);
על מנת להקצות מספיק מקום יש להוסיף +1 עבור null terminator '\0'

3. assert(out)
דרישת הפונקציה היא להחזיר NULL במקרה של שגיאה, לפיכך עבור שגיאת זיכרון יש לבדוק באמצעות if ולהחזיר NULL.

4. for (int i=0; i<=times; i++){
התוכנית תרוץ times+1 פעמים במקום times. לכן צריך להיות תנאי i<times

5. יש להחליף את סדר השורות
```
out = out + LEN;
strcpy(out,s);
```
על פי הסדר הנוכחי out יתקדם מעבר למקום המוקצה לו בזיכרון ותתרחש חריגת זיכרון.

6. return out;
מכיוון שהפוינטר out התקדם באיטרציות, הערך המוחזר הוא לא הכתובת הנכונה, לכן יש לשמור את כתובת ההתחלה המתקבלת בmalloc באמצעות פוינטר נוסף.

התכנית המתוקנת:

```c
#include "stdlib.h"
#include "string.h"
#include "assert.h"

char* stringDuplicator(char* s, int times)
{
    assert(s);
    assert(times > 0);

    int len = strlen(s);
    char* out = malloc(len*times + 1);
    if (!out)
    {
        return (NULL);
    }
    char* p_retval = out;

    for (int i=0; i<times; i++){
        strcpy(out,s);
        out = out + len;
    }

    return p_retval;
}
```