2.1
סעיף א'

```c
1  #include "stdlib.h"
2  #include "string.h"
3  #include "assert.h"
4  char* stringDuplicator(char* s, int times){
5  assert(!s);
6  assert(times > 0);
7  int LEN = strlen(s);
8  char* out = malloc(LEN*times);
9  assert(out);
10 for (int i=0; i<=times; i++){
11 out = out + LEN;
12 strcpy(out,s);
13 }
14 return out;
15 }
```

שורה	בעיה	
4	קונבנציה 1	שם למחזורת המתקבלת מהשתמש 's' אינו קיצור חוקי ע"פ קובץ הקונבנציות, הוא גם אינו מעיד על המחזרות ולא נותן לנו שום ערך מוסף, אם כבר הוא גורם לבלבול.
7	קונבנציה 2	שם המשתנה LEN אינו חוקי ע"פ קובץ הקונבנציה משום שהוא לא camelCase.
5	תוכנית 1	הassert הוא הפוך, הוא צריך לבדוק האם המחזורת שהתקבלה היא לא NULL וכעת הוא בודק את ההפך.
8	תוכנית 2	בבקשת הזיכרון לא נעשית התחשבות בתו האחרון \0 של המחרוזת שנרצה לשמור, strlen() אינו מחשב את התו הזה.
9	תוכנית 3	נעשית בדיקה האם הmalloc אכן הצליח והקצה זיכרון בעזרת assert, זהו שימוש לא נכון בassert אשר אינו יעבוד כאשר הקוד יהיה בPROD (השורה הזאת בכלל לא תהיה "קיימת") ולא נעשית שום בדיקה אמיתית להאם הצלחנו או לא הצלחנו להקצות זיכרון עם הmalloc מה שעלול בהמשך להוביל לנסיון גישה לא חוקי לזיכרון ולקריסת התוכנית.
9	תוכנית 3	לא נעשה שום פעולה אם הmalloc לא הצליח, אנחנו צריכים בתור המפתחים להחליט מה לעשות בנידון האם לבקש שוב פעם זיכרון או לצאת מהתוכנית אבל להמשיך את התוכנית סביר מאוד שתהיה שגיאה בזמן הריצה כשננסה לגשת לזיכרון שלא קיבלנו.
10	תוכנית 4	בלולאה יש חזרה אחת יותר מדי צריך להיות קטן ממש במקום קטן שווה או להתחיל את האינדקס של i בתור 1 ולא 0.
11-12	תוכנית 5	השורות הפוכות, קודם כל צריך להעתיק את המחזרות למקום בזיכרון ורק אז לקדם את המצביע קדימה.
14	תוכנית 6	לאחר שינוי הכתובת של המצביע out אנחנו מחזירים את המצביע למילה האחרונה המחרוזת שהכפלנו, אנחנו צריכים להחזיר את הערך הראשוני של המצביע out כדי שהמשתמש יוכל לקבל את כל המחרוזת שהכפלנו.

קוד מתוקן
2.2
סעיף ב'

כל הבעיות שנרשמו לעיל תוקנו וגם שמות המשתנים שונו כדי שיהיו יותר אינדוקטיביים לקריאה, כמו כן הקוד סודר מבחינת ההזחה שלו.

```
#include <stdlib.h>
#include <string.h>
#include <assert.h>

char *stringDuplicator(char *stringToDuplicate, int numberOfDuplications)
{

    assert(stringToDuplicate);
    assert(numberOfDuplications > 0);

    int lengthOfString = strlen(stringToDuplicate);
    char *duplicatedString = malloc(lengthOfString * numberOfDuplications + 1);

    if(duplicatedString == NULL){
        return(NULL);
    }

    for (int i = 1; i <= numberOfDuplications; i++)
    {
        strcpy(duplicatedString, stringToDuplicate);
        duplicatedString = duplicatedString + lengthOfString;
    }

    return duplicatedString - (numberOfDuplications * lengthOfString);
}
```