סעיף א':

שגיאות קוד:

1. ה- assert הראשון צריך להיות ללא סימן קריאה על מנת לבדוק האם המחרוזת אינה ריקה. בקוד המקור התוכנית תפסק אם המחרוזת אינה ריקה.
2. השימוש ב- assert השני שגוי, אין להשתמש ב - assert על מנת לוודא תקינות קלט.
3. חסרה הקצאת ביט נוסף ל- "\0" בסוף המחרוזת החדשה שתיווצר.
4. לאחר הקצאת הזיכרון למחרוזת החדשה ישנו שימוש לא נכון ב assert על מנת לוודא הקצאת זיכרון, כאשר התוכנית תפעל ללא הדגל DNDEBUG הפקודה assert לא תפעל ולכן לא יהיה וידא של הצלחת הקצאת הזיכרון, בנוסף לכך צריך לבדוק שהערך שהתקבל מ- malloc אינו NULLf.
5. בלולאת ה- for בתנאי העצירה יש להוריד את סימן השווה ולהשאיר רק i<times , אם הספירה מתחילה מ-0 (כמו שהיא בפונקציית המקור) אז מוסיפים 1 למספר הפעמים שהלולאה תרוץ ולכן כאשר נעצור במספר הקטן ב-1 מ- times הלולאה תרוץ מספר פעמים המתאים לדרישה ולפרמטרים המסופקים.
6. בכל איטרציה של לולאת ה-for מקדמים את המצביע ל-out ולכן כשמחזירים אותו מוחזר מהפונקציה מצביע למיקום של ההעתקה האחרונה שהתבצעה ולכן תמיד out יצביע לשכפול האחרון שהתבצע במקום שיוחזר מצביע לתחילת המחרוזת המקורית (ובכך יכללו גם ההעתקות).

שגיאות קונבנציה:

1. לפי קונבנציית מתן השמות למשתנים s אינו קיצור מקובל לstring ויש להשתמש בstr
2. לפי קונבנציית מתן השמות למשתנים LEN אינו שם מקובל, יש להשתמש באותיות קטנות לשמות משתנים, אלא אם מדובר באות ראשונה של מילה חדשה (עפ"י שיטת camelCase)
3. לפי קונבנציית מבנה הקוד, חובה להשתמש בהזחות לכל בלוק המוקף בסוגריים – בקטע הקוד הנמצא בתוך לולאת ה-for חסרה הזחה לשתי השורות שבלולאה.

סעיף ב':

קוד מתוקן:

```
char* stringDuplicator(char* str, int times) {
    assert(str);
    int len = strlen( s: str);
    char* out = (char*) malloc( size: len*times +1);
    if (out == NULL) {
        printf("Can't allocate memory\n");
        return NULL;
    }
    int j = 0;
    for (int i=0; i < times; i++)
    {
        strcpy(out + j ,str);
        j+=len;
    }
    return out;
}
```

---

*j משמש כמשתנה מקדם אינדקס