חלק יבש

דניאל בלוקון 318740768

2.1.1

1. קונבנצית שמות לפונקציות – צריך להיות פועל, צריך להיות camelCase. דוג' - duplicateString
2. חסרות הזחות בfor
3. הסוגר הפותח של הפונקציה לא נמצא בשורה נפרדת
4. LEN הוא משתנה ולכן צריך להיות בpascal_case, במקרה הזה פשוט len.

1. הדרישה היא שהפונקציה תחזיר NULL בכישלון – שלושת המקומות בהם יש assert צריכים להיות מוחלפים בבדיקה והחזרת NULL בכשלון.
2. הפונקציה strlen מקבלת כפרמטר את המחרוזת (* char), יש להוריד את הdereferencer.
3. הקצאת הזכרון למשתנה out לא מקצה מספיק מקום לnull terminator, להוסיף +1. (יש שיגידו שצריך לעשות גם sizeof ל char או out*... אבל זה תמיד 1).
4. כבר בהרצה הראשונה של הלולאה הערך של out בהעתקה הוא out+len, כלומר הוא יתחיל ממיקום אחד קדימה. נרצה להעביר לstrcpy את len*i + out או לחילופין להעתיק את המחרוזת הראשונה לפני הלולאה, אך אז אנחנו עדיין נתקלים בבעיה הבאה שהיא –
5. המצביע שאנחנו מחזירים הוא לא לתחילת המחרוזת אלא ל"תת מחרוזת" האחרונה במחרוזת הכפלית. נוכל לפתור את זה אם נעביר לstrcpy את המשתנה + ההזזה לעומת השמה בתוך המשתנה out.

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

char *duplicateString(char *s, int times)
{
    if (!s || times > 0)
        return NULL;
    int len = strlen(s);
    char *out = malloc(len * times + 1);

    if (!out)
        return NULL;

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

    return out;
}
```

```
void ListDelete(Node list)
{
    Node next = NULL;
    while (list)
    {
        next = list->next;
        free(list);
        list = next;
    }
}

ErrorCode mergeSortedLists(Node list1, Node list2, Node *mergedOut)
{
    if (!list1 || !list2)
        return NULL_ARGUMENT;
    *mergedOut = malloc(sizeof(*mergedOut));

    if (!*mergedOut)
        return MEMORY_ERROR;

    Node iterator = *mergedOut;
    do
    {
        if (!list1)
        {
            iterator->x = list2->x;
            list2 = list2->next;
        }
        else if (!list2)
        {
            iterator->x = list1->x;
            list1 = list1->next;
        }
        else if (list1->x < list2->x)
        {
            iterator->x = list1->x;
            list1 = list1->next;
        }
        else
        {
            iterator->x = list2->x;
            list2 = list2->next;
        }
        iterator->next = malloc(sizeof(*(iterator->next)));
        if (!iterator->next)
        {
            ListDelete(*mergedOut);
            return MEMORY_ERROR;
        }
        iterator = iterator->next;

    } while (list1 || list2);

    free(iterator);

    return SUCCESS;
}
```