תרגיל בית 1- חלק יבש

מגישות: שחר שטרית, 316371798
שחר כהן, 313553158

חלק 1- מיזוג רשימות ממוינות

```
#include <stdlib.h>
#include <stdio.h>
#include <stdbool.h>

typedef struct node_t {
    int x;
    struct node_t *next;
} *Node;

typedef enum {
    SUCCESS=0,
    MEMORY_ERROR,
    UNSORTED_LIST,
    NULL_ARGUMENT,
} ErrorCode;

int getListLength(Node list);
bool isListSorted(Node list);

Node createNode(int value) {
    Node new_node = malloc(sizeof(*new_node));
    new_node->x = value;
    new_node->next = NULL;
    return new_node;
}

void destroy_list(Node node) {
    while(node != NULL) {
        Node to_delete = node;
        node = node->next;
        free(to_delete);
    }
}

Node findSmallerValue(Node list1, Node list2) {
    return list1->x <= list2->x ? list1 : list2;
}

void copyTail(Node list, Node head, Node current_node) {
    while(list) {
        Node new_node = createNode(list->x);
        if(new_node == NULL) {
            printf("Memory allocation failed.\n");
            destroy_list(head);
            exit(1);
        }
```

```
        current_node->next = new_node;
        list = list->next;
        }
}

Node mergeSortedLists(Node list1, Node list2, ErrorCode* error_code) {

    if(!isListSorted(list1) || !isListSorted(list2)) {
        *error_code = UNSORTED_LIST;
        return NULL;
    }
    if(getListLength(list1) == 0 || getListLength(list2) == 0) {
        *error_code = NULL_ARGUMENT;
        return NULL;
    }
    //Now both lists should be valid

    Node current_node = createNode((findSmallerValue(list1, list2))->x);
    if(findSmallerValue(list1, list2) == list1)
            list1 = list1->next;
        else
            list2 = list2->next;
    //Pointing to next element, for the next comparison
    Node head = current_node;

    while((list1 != NULL) && (list2 != NULL)) {
        Node new_node = createNode((findSmallerValue(list1, list2))->x);
        if(new_node == NULL) {
            printf("Memory allocation failed.\n");
            destroy_list(head);
            exit(1);
        }
        current_node->next = new_node;
        if(findSmallerValue(list1, list2) == list1)
            list1 = list1->next;
        else
            list2 = list2->next;
        //Pointing to next element, for the next comparison
        current_node = current_node->next;
    }

    //Out of the while loop, now copying the numbers that are left
    if(list2 == NULL) {
        copyTail(list1, head, current_node);
    }
    else {
        copyTail(list2, head, current_node);
    }

    *error_code = SUCCESS;
    return head;
}
```

חלק 2- מציאת שגיאות

שגיאות הנכונות:

1) אין $dereference$ (*) לפני הגדרת המשתנה x.
2) הפונקציה מדפיסה את המחרוזת במקרה שאורכה זוגי במקום אי זוגי, ומדפיסה את המחרוזת ההופכית במקרה שאורכה אי זוגי במקום זוגי.
3) חסר -1 במיקום של השמת האותיות במחרוזת ההופכית.
4) בהתנהגותה הרצויה של הפונקציה מצוין כי "אם x הוא מצביע תקין...", אך אין בקוד כל בדיקה לכך.
5) אין וידוא הצלחה של הקצאת הזיכרון ע"י $malloc$.
6) הוספת +1 בהגדרת ה$malloc$ על מנת לאפשר מקום ל"\0".

שגיאות התכנות הנכון:

7) העברת המשנה new_str סמוך למקום הגדרתו.
8) התנאי if מופיע פעמיים, ללא $else$. אין צורך ב-if השני, כיוון ש-$else$ היה מתאר את כל שאר המקרים.
9) המשתנה x מתאר את אורך המחרוזת, אך אינו בעל שם אינפורמטיבי.
השארנו את שם המשתנה ב-x, אך שם מתאים יותר היה למשל "$length$".
10) הוספת $const$ לפני $char*$ כהכרזה על כך שה- char* לא אמור להשתנות ע"י המשתמש.
11) המשתנה $str2$ אינו בעל שם אינפורמטיבי, לכן שינינו אותו ל-new_str.

קוד מתוקן:

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

char* foo(const char* str, int* x) { //10

    int i;
    if(x == NULL) { //4
        printf("Pointer is not valid.");
        exit(1);
    }
    else {
        *x = strlen(str); //1, 9
        char* new_str; //7, 11
        new_str = malloc(sizeof(*x)+1); //6
        if(str == NULL) {//5
            printf("Memory allocation failed.");
            exit(1);
        }
        else {
            for (i = 0; i < *x; i++)
                new_str[i] = str[*x - i - 1]; //3
            if (*x % 2 == 0) //2
                printf("%s", new_str);
            else //8
                printf("%s", str);
            return new_str;
        }
    }
}
```