מבוא לתכנות מערכות

תרגיל בית 1

חלק יבש

מיזוג רשימות מקושרות ממוינות:

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <string.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 mergeSortedLists(Node list1, Node list2, ErrorCode* error_code);
void destroyList(Node ptr);
Node createNode(int value);
Node addNodeToList(Node current, int value, Node head, ErrorCode*
error_code);

//Implemention:
Node createNode(int value)
{
    Node node = malloc(sizeof(*node));
    if (node == NULL)
    {
        return NULL;
    }
    node->x = value;
    node->next = NULL;
    return node;
}

void destroyList(Node ptr)
{
    while (ptr != NULL)
    {
        Node toDelete = ptr;
        ptr = ptr->next;
        free(toDelete);
    }
}
```

```
Node addNodeToList(Node current, int value, Node head, ErrorCode*
error_code)
{
    current->next = createNode(value);
    if (current->next == NULL)
    {
        *error_code = MEMORY_ERROR;
        destroyList(head);
        return NULL;
    }
    return current->next;
}

Node mergeSortedLists(Node list1, Node list2, ErrorCode* error_code)
{
    if (list1 == NULL || list2 == NULL)
    {
        *error_code = NULL_ARGUMENT;
        return NULL;
    }
    if (isListSorted(list1) != true || isListSorted(list2) != true)
    {
        *error_code = UNSORTED_LIST;
        return NULL;
    }

    Node head;

    if (list1->x < list2->x)
    {
        head = createNode(list1->x);
        if (head == NULL)
        {
            *error_code = MEMORY_ERROR;
            return NULL;
        }
        list1 = list1->next;
    }
    else
    {
        head = createNode(list2->x);
        if (head == NULL)
        {
            *error_code = MEMORY_ERROR;
            return NULL;
        }
        list2 = list2->next;
    }
    Node current = head;
    while (list1 != NULL && list2 != NULL)
    {
        if (list1->x < list2->x)
        {
            current = addNodeToList(current, list1->x, head, error_code);
```

```c
            if (current == NULL)
            {
                return NULL;
            }
            list1 = list1->next;
        }
        else
        {
            current = addNodeToList(current, list2->x, head, error_code);
            if (current == NULL)
            {
                return NULL;
            }
            list2 = list2->next;
        }
    }

    while (list1 != NULL)
    {
        current = addNodeToList(current, list1->x, head, error_code);
        if (current == NULL)
        {
            return NULL;
        }
        list1 = list1->next;
    }

    while (list2 != NULL)
    {
        current = addNodeToList(current, list2->x, head, error_code);
        if (current == NULL)
        {
            return NULL;
        }
        list2 = list2->next;
    }
    *error_code = SUCCESS;
    return head;
}
```

מציאת שגיאות:

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

//error 1 - we need to give the function a meaningfull name
char* reverseEvenString(char* str, int* x)
//error 2 - we need to give the { of a function its own line
{
//error 3 - we need to check if str is not NULL
    if (str == NULL) {
        return NULL;
    }
//error 4 - we need to check check if x is not NULL
    if (x == NULL) {
        return NULL;
    }
    char* str2;
    int i;
//error 5 - we need to dereference the pointer to put the value in it
    *x = strlen(str);
//error 6 - we need to multiply the memory needed with sizeof char (conventions
error)
//error 7 - needs to consider the null space.
    str2 = malloc((sizeof(char)) * ((*x) + 1));
    if (str2 == NULL) {
//error 8 - we need to make sure malloc succeded
        return NULL;
    }
//error 9 - we need to make an indent line and use {}
    for (i = 0; i < *x; i++) {
//error 10 - don't want to copy the null space: need to add '-1' to the index we
access.
        str2[i] = str[*x - i - 1];
    }
//error 11 - needs to add the null space to the end of the string
    str2[*x] = '\0';
// error 12 - needs to print the origin string if the length is odd, changed the
condition.
    if (*x % 2 != 0) {
        printf("%s", str);
    }
// error 13 - needs to print the reverse string if the length is even, changed the
condition.
    if (*x % 2 == 0) {
        printf("%s", str2);
    }

    return str2;
}
```