תרגיל בית 1: מבוא לתכנות מערכות:

חלק יבש 2.1:

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


//STRUCTURES DECLARATIONS
typedef struct node_t
{
    int x;
    struct node_t *next;
} *Node;


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

//HELPER FUNCTIONS DECLARATIONS
int getListLength(Node list);
bool isListSorted(Node list);
int verifyAllocationMemory (Node node, ErrorCode* errorCode);
void mergeTheFirstElement (Node list1, Node list2, Node merge_out);
void mergeTheRest (Node list1, Node list2, Node merge_out, ErrorCode* errorCode);
void mergeOneList (Node list, Node merge_out, ErrorCode* errorCode);
```

```c
//PRINCIPAL FUNCTION
Node mergeSortedLists (Node list1, Node list2, ErrorCode* errorCode)
{
    //checking if one of the list is empty
    if(getListLength(list1)==0 || getListLength(list2)==0)
    {
        *errorCode=NULL_ARGUMENT;
        return NULL;
    }
    //checking if the two lists to merge are already sorted
    if(isListSorted(list1)==false || isListSorted(list2)==false)
    {
        *errorCode=UNSORTED_LIST;
        return NULL;
    }
    Node merge_out=malloc(sizeof(*merge_out));
    verifyAllocationMemory(merge_out, errorCode);

    //compare the first element of each list and merge it into the new list
    mergeTheFirstElement(list1, list2, merge_out);
```

```c
    //merging all the lists
    while (list1->next != NULL || list2->next != NULL)
    {
        merge_out->next=malloc(sizeof(*merge_out->next));
        verifyAllocationMemory(merge_out->next, errorCode);
        if (list1->x <= list2->x)
        {
            merge_out->x = list1->x;
            merge_out->next = NULL;
            list1 = list1->next;
            merge_out = merge_out->next;
        }
        else
        {
            merge_out->x = list2->x;
            merge_out->next = NULL;
            list2 = list2->next;
            merge_out = merge_out->next;
        }
    }
    //merge the rest of the biggest list
    mergeTheRest(list1, list2, merge_out, errorCode);
    *errorCode=SUCCESS;
    return merge_out;
}
```

```c
int verifyAllocationMemory (Node node, ErrorCode* errorCode)
{
    if(node==NULL)
    {
        *errorCode=MEMORY_ERROR;
        return *errorCode;
    }
}

void mergeTheFirstElement (Node list1, Node list2, Node merge_out)
{
    if (list1->x <= list2->x)
    {
        merge_out->x = list1->x;
        merge_out->next = NULL;
        list1 = list1->next;
        merge_out = merge_out->next;
    }
    else
    {
        merge_out->x = list2->x;
        merge_out->next = NULL;
        list2 = list2->next;
        merge_out = merge_out->next;
    }
}
```

```c
void mergeOneList (Node list, Node merge_out, ErrorCode* errorCode)
{
    merge_out->next=malloc(sizeof(*merge_out->next));
    verifyAllocationMemory(merge_out->next, errorCode);
    merge_out->x = list->x;
    merge_out->next = NULL;
    list = list->next;
    merge_out = merge_out->next;
}

void mergeTheRest (Node list1, Node list2, Node merge_out, ErrorCode* errorCode)
{
    if (list1->next == NULL)
    {
        while (list2->next != NULL)
        {
            mergeOneList(list2, merge_out, errorCode);
        }
    }

    else
    {
        while(list1->next!=NULL)
        {
            mergeOneList(list1, merge_out, errorCode);
        }
    }
}
```

חלק יבש 2.2: מציאת שגיאות

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   #include <string.h>
4
5   char* foo(char* str, int* x) {
6       char* str2;
7       int i;
8       x = strlen(str);
9       str2 = malloc(*x);
10      for (i = 0; i < *x; i++)
11          str2[i] = str[*x - i];
12      if (*x % 2 == 0) {
13          printf("%s", str);
14      }
15      if (*x % 2 != 0)
16      {
17          printf("%s", str2);
18      }
19      return str2;
20  }
```

שגיאות נכונות ממימוש:

- שורה 4 - צריך להוסיף * לפני ה- x ← *x=strlen(str)
- צריך להחליף בין 2 הפקודות ב- foo: בין 2 ה- if (מדפיס את המחרוזת ההפוכה אם אורכה זוגי).
- צריך להוסיף 1 בשורה 4 (בשביל ה- "\0")
- בפונקציה ה- foo כאשר i=0 מתקיים str2[0]=str[*x] צריך אומרת הוא תפס את ה- \0 הראשון של המחרוזת שמחכה.

מה יוצא מכלל קבוע כרגע:

- לשנות את שם הפונקציה לשם שם משמעותי.
- צריך לרדת שורה לפני שפותחים את הסוגר המסולסל.
- צריך להוסיף סוגרים מסולסלים בפקודה של ה- foo.
- צריך להגדיר את המספר 2 ב- define.

הקוד הנכון:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define TWO 2
#define ZERO 0
#define END_OF_STRING 1 //the '/0' element

char*  reverseIfEven (char * string_before_reverse, int* length_of_string)
{
    char* string_after_inverse;
    *length_of_string = (int)strlen(string_before_reverse)+ END_OF_STRING;
    string_after_inverse=malloc( _Size: sizeof(char)*(*length_of_string));
    if(string_after_inverse==NULL)
    {
        return NULL;
    }
    int i;
    for(i=0 ; i<(*length_of_string) ; i++)
    {
        string_after_inverse[i] = string_before_reverse[(*length_of_string)-i-END_OF_STRING];
    }
    if((*length_of_string)%TWO != ZERO)
    {
        printf( _Format: "%s",string_before_reverse);
    }
    if((*length_of_string)%TWO == ZERO)
    {
        printf( _Format: "%s",string_after_inverse);
    }
    return string_after_inverse;
}
```