תרגיל בית 1: מבוא לתכנות מערכות:

<u>חלק יבש 2.1:</u>

```
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
typedef struct node_t
   struct node_t *next;
} *Node;
NULL_ARGUMENT,
} ErrorCode;
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);
```

```
//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);
```

```
//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->next = NULL;
        list1 = list1->next;
        merge_out = merge_out->next;
    }
    else
    {
        merge_out->next = NULL;
        list2 = list2->x;
        merge_out->next;
        merge_out->next;
        merge_out->next;
        merge_out->next;
        merge_out = merge_out->next;
    }
}
//merge the rest of the biggest list
mergeTheRest(list1, list2, merge_out, errorCode);
*errorCode=SUCCESS;
return merge_out;
}
```

```
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->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;
    }
}
```

```
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;

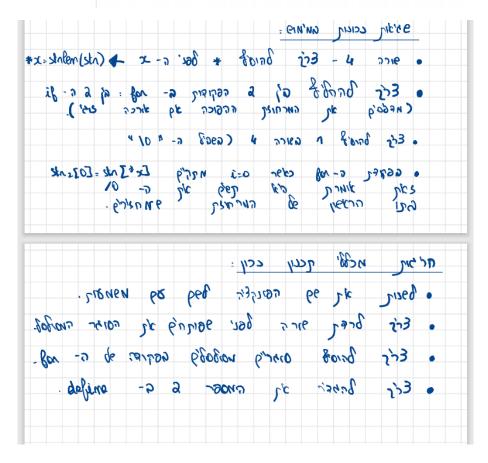
### Mode list1, Node list2, Node merge_out, ErrorCode* errorCode)

{
    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

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



הקוד הנכון: