מת"מ תרגיל בית 1

מגישים: זיו דמיר-206606709
בועז פרידר- 206699522

בקובץ : קובץ מתוקן של foo.

merge sorted list.-

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


char* foo(char* str, int* x) {

    char* str2;
    int i;
    if(x==NULL || str == NULL)//1-need to check the addres of X, and if it valid save the len !
    {
       return NULL;
    }
    *x = (int)strlen(str); //2- add pointer to X  and casting the value to int from"size_t"
    str2 =(char*) malloc(sizeof(char)*(*x+1)); //plus 1 for '/0' , 3-fixed the way we created the malloc.

    if(str2==NULL) //4 - add check for mallocd .
    {
        free(str2);
        return NULL;
    }
   int len=*x;
    for (i = 0; i <= len; i++) // 5- adding "=" because we have the special last char.
        str2[i] = str[len- i-1]; // 6- adding "-1"

    if (len % 2 != 0) { // 7- fix to "!=" because if the len is odd we need to pring the original string
        printf("%s", str);
    }

    if (len % 2 == 0) // 8-fix to "==" , because if the len is even we need to pring the oppsite string.
    {
        printf("%s", str2);
    }
    return str2;
}


```

```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);
int compareElements(int x1,int x2);
void DestroyList(Node list);
void AddElement(Node list,int x);
Node CreateElement(int x);
//returns 0,if x1=x2,x1>x2 will return positive  number,x1<x2 will return negative number.
int getListLength(Node list) {
    int len=0;
    Node i=list;
    while(i){
        len++;
        i=i->next;
    }
    return len;
}

Node CreateElement(int y)
{
    Node new=(Node)malloc(sizeof(*new));
    if(!new) return NULL;
    new->x=y;
    return new;
}


void DestroyList(Node list)
{
    if(list == NULL) return;
    while(list!= NULL )
    {
      Node tmp=list;
      list=list->next;
       free(tmp);
       tmp=NULL;
    }
}
int compareElements(int x1,int x2)
{
    return (x1-x2);
}
Node mergeSortedLists(Node list1, Node list2, ErrorCode* error_code)
```

```
{
    int size1, size2;
    size1=getListLength(list1),size2=getListLength(list2);
    if(size1 == 0)
    {
         *error_code=SUCCESS;
        return list2;
    }
    if(size2 == 0)
    {
        *error_code=SUCCESS;
        return list1;
    }
    Node merged_list=(Node)malloc(sizeof(*merged_list));
    if(!merged_list)
    {
        *error_code=MEMORY_ERROR;
        return NULL;
    }
//merge
    int result=0;
    Node tmp1=list1;
    Node tmp2=list2;
    Node tmp3=merged_list;//will save the current "head" of list1,list2,list3.
    while( list1!= NULL && list2!= NULL )
    {
        result=compareElements(list1->x,list2->x);//are there any cases for x1,x2?
        if(result <= 0)
        {
            merged_list->x=list1->x;
            list1=list1->next;
        }
        //advancing of merged_list will happen in the end.
        //else- result >0
        else
        {
            merged_list->x=list2->x;
            list2=list2->next;
        }
        //now malloc the next Node for merged_list.
        merged_list->next=(Node)malloc(sizeof(*merged_list->next));
        if(!merged_list->next)
        {
            //free all things that was malloced in list.
            DestroyList(tmp3);
            *error_code=MEMORY_ERROR;
            return NULL;
        }
        merged_list=merged_list->next;
    }
//now after one of lists of list1,list2 went empty,we iterate throught each one and
just insert the elements there..
    while(list1!=NULL)
    {
        //we assume the first iteration is malloced
        merged_list->x=list1->x;
        list1=list1->next;
        if(list1!=NULL)
        {
        merged_list->next=(Node)malloc(sizeof(*merged_list->next));
```

```
119         //we know for fact(or else we wont be here), that one of the lists is
    null.in this case list 2 is null
120         //the last element makes us problems in compilation
121         //so we check whether or not there exists antoer element in list1,if yes,we
    malloc,else,we dont
122         if(!merged_list->next)
123           {
124             DestroyList(tmp3);
125             *error_code=MEMORY_ERROR;
126             return NULL;
127           }
128            merged_list=merged_list->next;
129         }
130      }
131     while(list2!=NULL)
132     {
133         //we assume the first iteration is malloced
134         merged_list->x=list2->x;
135         list2=list2->next;
136          //we know for fact(or else we wont be here), that one of the lists is
    null.in this case list 1 is null
137         //the last element makes us problems in compilation
138         //so we check whether or not there exists antoer element in list2⊓if yes,we
    malloc,else,we dont
139       if(list2!=NULL)
140       {
141         merged_list->next=(Node)malloc(sizeof(*merged_list->next));
142         if(!merged_list->next)
143          {
144             DestroyList(tmp3);
145             *error_code=MEMORY_ERROR;
146             return NULL;
147          }
148         merged_list=merged_list->next;
149       }
150     }
151     *error_code=SUCCESS;
152     merged_list=tmp3;
153     list1=tmp1;
154     list2=tmp2;
155     return merged_list;
156 }
```