Question 1:

```
ErrorCode chainLinkList(Node *source, Node list)
{
    if (source == NULL || list == NULL)
    {
        return SUCCESS;
    }
    // Go through the current list.
    Node temp = (*source);
    while (list != NULL)
    {
        // Allocate next node.
        Node new_node = malloc(sizeof(*temp));
        if (new_node == NULL)
        {
            return MEMORY_ERROR;
        }
        new_node->x = list->x;
        temp->next = new_node;
        list = list->next;
        temp = new_node;
    }
    temp->next = NULL;
    return SUCCESS;
}
```

```c
ErrorCode mergeSortedLists(Node list1, Node list2, Node *merged_out)
{
    if (list1 == NULL || list2 == NULL)
    {
        return EMPTY_LIST;
    }
    // Allocate new node.
    Node node = malloc(sizeof(*node));
    if (node == NULL)
    {
        return MEMORY_ERROR;
    }
    *merged_out = node;
    // Get first.
    Node temp = node;
    if (list1->x > list2->x)
    {
        // Set from list 2.
        temp->x = list2->x;
        list2 = list2->next;
    }
    else
    {
        temp->x = list1->x;
        list1 = list1->next;
    }
    while (list1 != NULL && list2 != NULL)
    {
        // Allocate new temp node.
        Node new_node = malloc(sizeof(*new_node));
        if (new_node == NULL)
        {
            *merged_out = NULL;
            return MEMORY_ERROR;
        }
        temp->next = new_node;
        if (list1->x > list2->x)
        {
            // Set from list 2.
            new_node->x = list2->x;
            list2 = list2->next;
        }
        else
        {
            new_node->x = list1->x;
            list1 = list1->next;
        }
        temp = temp->next;
    }

    if (chainLinkList(&temp, list1) == MEMORY_ERROR || chainLinkList(&temp, list2) == MEMORY_ERROR)
    {
        *merged_out = NULL;
        return MEMORY_ERROR;
    }
    return SUCCESS;
}
```

Question 2:

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

char *stringDuplicator(char *s, int times) {
 assert(!s);
 assert(times > 0);
 int LEN = strlen(s);
 char *out = malloc(LEN * times);
 assert(out);
 for (int i = 0; i < times; i++) {
 out = out + LEN;
 strcpy(out, s);
 }
 return out;
}

//code conventions mistakes:
//1.	LEN must be in small letters (len).
//2.	"0" must be defined as minimum times value (#define MIN 0).
//3.	in malloc the vriable size should be spicified(sizeof(char)).
//4.	the srting variable name "s" -> "str".

//programming mistakes:
//1.	assert() use in line 6 -> assert(s!=NULL).
//2.	replace line 12 with 13 and 13 with 12.
//3.	the output pointer loses the initial address, a temporary pointer should
```

```
//                    be used and return the out variable.
//4.      in line 10 assert(out) -> assert(out!=NULL).

 //the fixed program:

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <assert.h>
#define MIN 0

char *stringDuplicator(char *str, int times) {
        assert(str!=NULL);
        assert(times > MIN);
        int len = strlen(str);
        char *out = malloc(len * times* sizeof(char));
        assert(out!=NULL);
        char* temp = out;
        for (int i = 0; i < times; i++) {
                strcpy(temp, str);
                temp = temp + len;
        }
        return out;
}
```