#include <stdio.h>  
#include <stdlib.h>  
  
// 结构体定义  
typedef struct {  
    int\* data;  // 整数数据  
    int size;   // 数据长度  
    int index;  // 当前索引  
} NestedIterator;  
  
// 主函数：扁平化嵌套列表  
int\* flatten(int\*\* nestedList, int nestedListSize, int\* nestedListColSize, int\* returnSize) {  
    // 计算扁平化后的结果长度  
    int totalSize = 0;  
    for (int i = 0; i < nestedListSize; i++) {  
        totalSize += nestedListColSize[i];  
    }  
  
    // 分配内存空间  
    int\* result = (int\*)malloc(totalSize \* sizeof(int));  
  
    // 使用栈来处理元素  
    int\* stack = (int\*)malloc(totalSize \* sizeof(int));  
    int top = -1;  
  
    // 将所有列表元素压入栈中  
    for (int i = nestedListSize - 1; i >= 0; i--) {  
        for (int j = nestedListColSize[i] - 1; j >= 0; j--) {  
            stack[++top] = nestedList[i][j];  
        }  
    }  
  
    // 弹出栈顶元素并处理  
    int index = 0;  
    while (top >= 0) {  
        int item = stack[top--];  
        if (item >= 0) {  
            result[index++] = item;  // 如果是整数，则添加到结果中  
        } else {  
            // 如果是列表，则将其元素逆序压入栈中  
            for (int i = nestedListColSize[-item - 1] - 1; i >= 0; i--) {  
                stack[++top] = nestedList[-item - 1][i];  
            }  
        }  
    }  
  
    \*returnSize = totalSize;  
    free(stack);  
  
    return result;  
}  
  
int main() {  
    int nestedList1[] = {1, 2, 3};  
    int nestedList2[] = {4, 5};  
    int nestedList3[] = {6};  
    int\* nestedList[] = {nestedList1, nestedList2, nestedList3};  
    int nestedListSize = 3;  
    int nestedListColSize[] = {3, 2, 1};  
  
    int returnSize = 0;  
    int\* flattenedList = flatten(nestedList, nestedListSize, nestedListColSize, &returnSize);  
  
    printf("Flattened List: ");  
    for (int i = 0; i < returnSize; i++) {  
        printf("%d ", flattenedList[i]);  
    }  
    printf("\n");  
  
    free(flattenedList);  
  
    return 0;  
}

#include <stdio.h>  
#include <stdlib.h>  
  
// 结构体定义  
typedef struct {  
    int\* data;  // 整数数据  
    int size;   // 数据长度  
    int index;  // 当前索引  
} NestedIterator;  
  
// 辅助函数：递归处理列表  
void flattenHelper(int\* arr, int size, int\* result, int\* index) {  
    for (int i = 0; i < size; i++) {  
        if (arr[i] >= 0) {  
            result[(\*index)++] = arr[i];  // 如果是整数，则添加到结果中  
        } else {  
            flattenHelper(arr + i + 1, arr[i], result, index);  // 递归处理列表  
            i += arr[i];  // 跳过已处理的列表元素  
        }  
    }  
}  
  
// 主函数：扁平化嵌套列表  
int\* flatten(int\*\* nestedList, int nestedListSize, int\* nestedListColSize, int\* returnSize) {  
    // 计算扁平化后的结果长度  
    int totalSize = 0;  
    for (int i = 0; i < nestedListSize; i++) {  
        totalSize += nestedListColSize[i];  
    }  
  
    // 分配内存空间  
    int\* result = (int\*)malloc(totalSize \* sizeof(int));  
    int index = 0;  
  
    // 递归处理列表  
    for (int i = 0; i < nestedListSize; i++) {  
        flattenHelper(nestedList[i], nestedListColSize[i], result, &index);  
    }  
  
    \*returnSize = totalSize;  
    return result;  
}  
  
int main() {  
    int nestedList1[] = {1, 2, 3};  
    int nestedList2[] = {4, 5};  
    int nestedList3[] = {6};  
    int\* nestedList[] = {nestedList1, nestedList2, nestedList3};  
    int nestedListSize = 3;  
    int nestedListColSize[] = {3, 2, 1};  
  
    int returnSize = 0;  
    int\* flattenedList = flatten(nestedList, nestedListSize, nestedListColSize, &returnSize);  
  
    printf("Flattened List: ");  
    for (int i = 0; i < returnSize; i++) {  
        printf("%d ", flattenedList[i]);  
    }  
    printf("\n");  
  
    free(flattenedList);  
  
    return 0;  
}