

[PI054] Removing All Occurrences of an Element



Base Code

Use the provided code as your base: **linkedList.h** ([view code](#) | [download code](#)) and **linkedList.c** ([view code](#) | [download code](#)), which contain the data structure definitions (including `Node` and `LinkedList`), prototype declarations, and their implementations as discussed in class. The code implements a singly linked list where the node values are of type `NodeInfo`, which in this problem corresponds to `int`. It includes the usual methods for adding and removing elements at the beginning or end, retrieving the size of the list, etc.

The Problem

Extend the linked list definition with a new function: **`void removeAll(NodeInfo value, LinkedList *list)`**, which **removes all occurrences of the value *value* from the list *list***. In this problem, `NodeInfo` is of type `int`.

Mooshak Submission

To submit in Mooshak, you must submit a program called **submit.c**, in which you `#include` the file `linkedList.h` and include your implementation of the **`removeAll`** function as required.

Mooshak will compile the files `linkedList.c`, `submit.c`, and `test.c` (provided below, containing the `main()` function).

```
#include "linkedList.h"

// Forward declaration of function to implement
void removeAll(NodeInfo x, LinkedList *l);

void printList2(LinkedList *list) {
    Node *curr= list->first;

    printf("{");
    while (curr != NULL) {
        printf("%d",curr->val);
        curr= curr->next;
        printf("%c",","}[curr==NULL]);
    }
}

void testInt(LinkedList *list) {
    printf("list = ");
    printList2(list);
    printf("[size=%d]\n", list->size);
    int value;
    scanf("%d", &value);
}
```

```
printf("removeAll(%d)",value);
removeAll(value, list);
printf("\nlist = ");
printList2(list);
printf(" [size=%d]\n", list->size);
printf("-----\n");
}

int main() {
    int ncases;
    int n;
    int value;
    LinkedList l1;


    scanf("%d",&ncases);           // number of cases
    for (int i=0; i<ncases; i++) {
        initList(&l1);
        scanf("%d",&n);           // number of values in the list
        for (int i=0; i<n; i++) {
            scanf("%d",&value);    // the values
            addLast(&l1, value);
        }
        // test
        testInt(&l1);             // test this case
    }
}
```

Exemplos de Input/Output

Lista inicial	Chamada	Estado da lista depois da chamada
list = {1,2,2,2,1,3,4,2,1}	list.removeAll(1)	list = {2,2,2,3,4,2}
list = {1,2,2,2,1,3,4,2,1}	list.removeAll(2)	list = {1,1,3,4,1}
list = {1,2,2,2,1,3,4,2,1}	list.removeAll(3)	list = {1,2,2,2,1,4,2,1}
list = {1,2,2,2,1,3,4,2,1}	list.removeAll(5)	list = {1,2,2,2,1,3,4,2,1}

 [Versão em Português](#) | [\[see english version\]](#)

[PI054] Removendo todas as ocorrências de um elemento



Código Base

Use como base o código **linkedList.h** ([ver código](#) | [\(download código\)](#) e **linkedList.c** ([ver código](#) | [\(download código\)](#) que contém as declarações das estruturas de dados (incluindo Node e LinkedList, funções protótipo e a sua implementação conforme dado nas aulas. O código implementa uma lista ligada simples em que os valores são do tipo NodeInfo que neste problema corresponde ao tipo int. Inclui os métodos habituais de adicionar e remover um elemento início ou no final,

devolver o tamanho, etc.

O problema

Acrescente à definição de listas ligadas uma nova função **void removeAll(NodeInfo value, LinkedList *list)** que **remove da lista *list* todas as ocorrências do valor *value***. Neste problema, o NodeInfo é um inteiro.

Submissão no Mooshak

Para submeter no Mooshak, deverá submeter um programa **submit.c**, no qual faça o "include" do ficheiro `linkedList.h` e inclua a sua implementação da função **removeAll** como pedido.

O Mooshak compilará os ficheiros `linkedList.c`, `submit.c` e o `test.c` que se segue com a função `main()`.

```
#include "linkedList.h"

// Forward declaration of function to implement
void removeAll(NodeInfo x, LinkedList *l);

void printList2(LinkedList *list) {
    Node *curr= list->first;

    printf("{");
    while (curr != NULL) {
        printf("%d",curr->val);
        curr= curr->next;
        printf("%c",",")[curr==NULL]);
    }
}

void testInt(LinkedList *list) {
    printf("list = ");
    printList2(list);
    printf(" [size=%d]\n", list->size);
    int value;
    scanf("%d", &value);
    printf("removeAll(%d)",value);
    removeAll(value, list);
    printf("\nlist = ");
    printList2(list);
    printf(" [size=%d]\n", list->size);
    printf("-----\n");
}

int main() {
    int ncases;
    int n;
    int value;
    LinkedList l1;

    scanf("%d",&ncases);           // number of cases
    for (int i=0; i<ncases; i++) {
        initList(&l1);
        scanf("%d",&n);           // number of values in the list
        for (int i=0; i<n; i++) {
            scanf("%d",&value);   // the values
            addLast(&l1, value);
        }
        // test
        testInt(&l1);             // test this case
    }
}
```

Exemplos de Input/Output

Lista inicial	Chamada	Estado da lista depois da chamada
list = {1,2,2,2,1,3,4,2,1}	list.removeAll(1)	list = {2,2,2,3,4,2}
list = {1,2,2,2,1,3,4,2,1}	list.removeAll(2)	list = {1,1,3,4,1}
list = {1,2,2,2,1,3,4,2,1}	list.removeAll(3)	list = {1,2,2,2,1,4,2,1}
list = {1,2,2,2,1,3,4,2,1}	list.removeAll(5)	list = {1,2,2,2,1,3,4,2,1}