Programação Imperativa

1º Ano – LCC/MIEF/MIEI

Questões 2ª Parte

Listas Ligadas

Considere o seguinte tipo para representar listas ligadas de inteiros

```
typedef struct lligada {
   int valor;
   struct lligada *prox;
} *LInt;
```

1. Apresente uma definição não recursiva da função `int length (LInt)` que calcula o comprimento de uma lista ligada.
   Give a non-recursive definition of the function ***`int length (LInt)`*** *which calculates the length of a linked list.*
   (`https://codeboard.io/projects/16161/summary`)

2. Apresente uma definição não recursiva da função `void freeL (LInt)` que liberta o espaço ocupado por uma lista.
   Give a non-recursive definition of the function ***`void freeL (LInt)`*** *that frees the space occupied by a list.*

3. Apresente uma definição não recursiva da função `void imprimeL (LInt)` que imprime no ecran os elementos de uma lista (um por linha).
   Give a non-recursive definition of the function ***`void imprimeL (LInt)`*** *which prints the elements of a list (one per line) on the screen.*

4. Apresente uma definição não recursiva da função `LInt reverseL (LInt)` que inverte uma lista (sem criar uma nova lista).
   Give a non-recursive definition of the function ***`LInt reverseL (LInt)`*** *which reverses a list (without creating a new list).*
   (`https://codeboard.io/projects/16243/summary`)

5. Apresente uma definição não recursiva da função `void insertOrd (LInt *, int)` que insere ordenadamente um elemento numa lista ordenada.
   Give a non-recursive definition of the function ***`void insertOrd (LInt *, int)`*** *which inserts an element into a sorted list.*
   (`https://codeboard.io/projects/16244/summary`)

6. Apresente uma definição não recursiva da função `int removeOneOrd (LInt *, int)` que remove um elemento de uma lista ordenada. Retorna 1 caso o elemento a remover não exista, 0 no outro caso.

   Give a non-recursive definition of the function `int removeOneOrd (LInt *, int)` which removes an element from an ordered list. It returns 1 if the element to remove does not exist, 0 otherwise.
   (`https://codeboard.io/projects/16245/summary`)

7. Defina uma função `void merge (LInt *r, LInt a, LInt b)` que junta duas listas ordenadas (`a` e `b`) numa única lista ordenada (`*r`).

   Define a function `void merge (LInt *r, LInt a, LInt b)` that joins two sorted lists (`a` and `b`) into a single sorted list (`*r`).
   (`https://codeboard.io/projects/16246/summary`)

8. Defina uma função `void splitQS (LInt l, int x, LInt *mx, LInt *Mx)` que, dada uma lista ligada `l` e um inteiro `x`, parte a lista em duas (retornando os endereços dos primeiros elementos da lista em `*mx` e `*Mx`): uma com os elementos de `l` menores do que `x` e a outra com os restantes. Note que esta função não deverá criar cópias dos elementos da lista.

   Define a function `void splitQS (LInt l, int x, LInt *mx, LInt *Mx)` which, given a linked list `l` and an integer `x`, splits the list in two (returning the addresses of the first elements of the list in `*mx` and `*Mx`): one with the elements of `l` smaller than `x` and the other with the rest. Note that this function should not create copies of the elements in the list.
   (`https://codeboard.io/projects/16247/summary`)

9. Defina uma função `LLig parteAmeio (LLig *l)` que, parte uma lista não vazia `*l` a meio. Se `x` contiver os elementos `1,2,3,4,5`, após a invocação `y=parteAmeio(&x)` a lista `y` deverá conter os elementos `1,2` e a lista `x` os restantes `3,4,5`.

   Define a function `LLig parteAmeio (LLig *l)` that breaks a non-empty list `*l` in half. If `x` contains the elements `1,2,3,4,5`, after invoking `y=parteAmeio(&x)` the list `y` should contain the elements `1,2` and the list `x` the remaining `3,4,5`.
   (`https://codeboard.io/projects/17392/summary`)

10. Apresente uma definição não recursiva da função `int removeAll (LInt *, int)` que remove todas as ocorrências de um dado inteiro de uma lista, retornando o número de células removidas.

    Give a non-recursive definition of the function `int removeAll (LInt *, int)` which removes all occurrences of a given integer from a list, returning the number of cells removed.
    (`https://codeboard.io/projects/16249/summary`)

11. Apresente uma definição da função `int removeDups (LInt *)` que remove os valores repetidos de uma lista (deixando apenas a primeira ocorrência).

    Give a definition of the function `int removeDups (LInt *)` which removes repeated values from a list (leaving only the first occurrence).
    (`https://codeboard.io/projects/16250/summary`)

12. Apresente uma definição da função `int removeMaiorL (LInt *)` que remove (a primeira ocorrência) o maior elemento de uma lista não vazia, retornando o valor desse elemento.

    Give a definition of the function `int removeMaiorL (LInt *)` which removes (the first occurrence of) the largest element from a non-empty list, returning the value of that element.
    (`https://codeboard.io/projects/16251/summary`)

13. Apresente uma definição não recursiva da função `void init (LInt *)` que remove o último elemento de uma lista não vazia (libertando o correspondente espaço).

    Give a non-recursive definition of the function ***`void init (LInt *)`*** *which removes the last element from a non-empty list (freeing the corresponding space).*
    (`https://codeboard.io/projects/16252/summary`)

14. Apresente uma definição não recursiva da função `void appendL (LInt *, int)` que acrescenta um elemento no fim da lista.

    Give a non-recursive definition of the function ***`void appendL (LInt *, int)`*** *which adds an element to the end of the list.*
    (`https://codeboard.io/projects/16253/summary`)

15. Apresente uma definição da função `void concatL (LInt *a, LInt b)` que acrescenta a lista `b` à lista `*a`.

    Give a definition of the function ***`void concatL (LInt *a, LInt b)`*** *which adds the list* `b` *to the list* ***`*a`****.*
    (`https://codeboard.io/projects/16254/summary`)

16. Apresente uma definição da função `LInt cloneL (LInt)` que cria uma nova lista ligada com os elementos pela ordem em que aparecem na lista argumento.

    Give a definition of the function ***`LInt cloneL (LInt)`*** *which creates a new linked list with the elements in the order in which they appear in the argument list.*

17. Apresente uma definição não recursiva da função `LInt cloneRev (LInt)` que cria uma nova lista ligada com os elementos por ordem inversa.
    Por exemplo, se a lista `l` tiver 5 elementos com os valores `[1,2,3,4,5]` por esta ordem, a invocação `cloneRev(l)` deve corresponder a uma nova lista com os elementos `[5,4,3,2,1]` por esta ordem.

    Give a non-recursive definition of the function ***`LInt cloneRev (LInt)`*** *which creates a new linked list with the elements in reverse order.*
    For example, if the list `l` *has 5 elements with the values* ***`[1,2,3,4,5]`*** *in this order,* ***`cloneRev(l)`*** *must correspond to a new list with the elements* ***`[5,4,3,2,1]`*** *in this order.*
    (`https://codeboard.io/projects/16256/summary`)

18. Defina uma função `int maximo (LInt l)` que calcula qual o maior valor armazenado numa lista não vazia.

    Define a function ***`int maximo (LInt l)`*** *that calculates the largest value stored in a non-empty list.*
    (`https://codeboard.io/projects/16257/summary`)

19. Apresente uma definição iterativa da função `int take (int n, LInt *l)` que, dado um inteiro `n` e uma lista ligada de inteiros `l`, apaga de `l` todos os nodos para além do `n`-ésimo (libertando o respectivo espaço). Se a lista tiver `n` ou menos nodos, a função não altera a lista.
    A função deve retornar o comprimento final da lista.

    Give an iterative definition of the function ***`int take (int n, LInt *l)`*** *which, given an integer* `n` *and a linked list of integers* `l`*, deletes all nodes after the* `n`*-th from* `l` *(freeing up the corresponding space). If the list has* `n` *or fewer nodes, the function does not change the list.*
    The function must return the final length of the list.
    (`https://codeboard.io/projects/16258/summary`)

20. Apresente uma definição iterativa da função `int drop (int n, LInt *l)` que, dado um inteiro `n` e uma lista ligada de inteiros `l`, apaga de `l` os `n` primeiros elementos da lista (libertando o respectivo espaço). Se a lista tiver `n` ou menos nodos, a função liberta a totalidade da lista. A função deve retornar o número de elementos removidos.

    Give an iterative definition of the function `int drop (int n, LInt *l)` which, given an integer `n` and a linked list of integers `l`, deletes the first `n` elements of the list from `l` (freeing the corresponding space). If the list has `n` or fewer nodes, the function frees the entire list.
    The function must return the number of elements removed.
    (`https://codeboard.io/projects/16259/summary`)

21. O tipo `LInt` pode ser usado ainda para implementar listas circulares. Defina uma função `LInt Nforward (LInt l, int N)` que, dada uma lista circular dá como resultado o endereço do elemento da lista que está `N` posições à frente.

    The type `LInt` can also be used to implement circular lists. Define a function `LInt Nforward (LInt l, int N)` which, given a circular list, computes the address of the list element that is `N` positions ahead.
    (`https://codeboard.io/projects/16260/summary`)

22. Defina uma função `int listToArray (LInt l, int v[], int N)` que, dada uma lista `l`, preenche o array `v` com os elementos da lista.
    A função deverá preencher no máximo `N` elementos e retornar o número de elementos preenchidos.

    Define a function `int listToArray (LInt l, int v[], int N)` which, given a list `l`, fills the array `v` with the elements of the list.
    The function should fill at most `N` elements and return the number of elements filled.
    (`https://codeboard.io/projects/16261/summary`)

23. Defina uma função `LInt arrayToList (int v[], int N)` que constrói uma lista com os elementos de um array, pela mesma ordem em que aparecem no array.

    Define a function `LInt arrayToList (int v[], int N)` that builds a list with the elements of an array, in the same order in which they appear in the array.
    (`https://codeboard.io/projects/16262/summary`)

24. Defina uma função `LInt somasAcL (LInt l)` que, dada uma lista de inteiros, constrói uma nova lista de inteiros contendo as somas acumuladas da lista original (que deverá permanecer inalterada).
    Por exemplo, se a lista `l` tiver os valores `[1,2,3,4]` a lista contruída pela invocação de `somasAcL (l)` deverá conter os valores `[1,3,6,10]`.

    Define a function `LInt somasAcL (LInt l)` which, given a list of integers, constructs a new list of integers containing the accumulated sums of the original list (which should remain unchanged).
    For example, if the list `l` has the values `[1,2,3,4]` the list built by invoking `somasAcL (l)` should contain the values `[1,3,6,10]`.
    (`https://codeboard.io/projects/16263/summary`)

25. Defina uma função `void remreps (LInt l)` que, dada uma lista ordenada de inteiros, elimina dessa lista todos os valores repetidos assegurando que o espaço de memória correspondente aos nós removidos é correctamente libertado.

    Define a function `void remreps (LInt l)` which, given an ordered list of integers, removes all repeated values from that list, ensuring that the memory space corresponding to the removed nodes is correctly

freed.
(`https://codeboard.io/projects/16264/summary`)

26. Defina uma função `LInt rotateL (LInt l)` que coloca o primeiro elemento de uma lista no fim. Se a lista for vazia ou tiver apenas um elemento, a função não tem qualquer efeito prático (i.e., devolve a mesma lista que recebe como argumento).
Note que a sua função não deve alocar nem libertar memória. Apenas re-organizar as células da lista.
Define a function `LInt rotateL (LInt l)` that puts the first element of a list at the end. If the list is empty or has only one element, the function has no practical effect (i.e., it returns the same list it receives as an argument).
Note that your function shouldn't allocate or free memory. It just re-organizes the cells in the list.
(`https://codeboard.io/projects/16265/summary`)

27. Defina uma função `LInt parte (LInt l)` que parte uma lista `l` em duas: na lista `l` ficam apenas os elementos das posições ímpares; na lista resultante ficam os restantes elementos. Assim, se a lista `x` tiver os elementos `[10,20,30,40,50,60]` a chamada `y = parte (x)`, coloca na lista `y` os elementos `[20,40,60]` ficando em `x` apenas os elementos `[10,30,50]`
Define a function `LInt parte (LInt l)` that splits a list `l` in two: in the list `l` only the elements in the odd positions remain; in the resulting list the remaining elements.
If the list `x` has the elements `[10,20,30,40,50,60]` the call `y = parte (x)`, puts the elements `[20,40,60]` in the list `y` with only the elements `[10,30,50]` in `x`.
(`https://codeboard.io/projects/16266/summary`)

Árvores Binárias

Considere o seguinte tipo para representar árvores binárias de inteiros

```
typedef struct nodo {
   int valor;
   struct nodo *esq, *dir;
} *ABin;
```

28. Apresente uma definição da função `int altura (ABin)` que calcula a altura de uma árvore binária.
Define the function `int altura (ABin)` which calculates the height of a binary tree.
(`https://codeboard.io/projects/16220/summary`)

29. Defina uma função `ABin cloneAB (ABin)` que cria uma cópia de uma árvore.
Define a function `ABin cloneAB (ABin)` that creates a copy of a tree.
(`https://codeboard.io/projects/16267/summary`)

30. Defina uma função `void mirror (ABin *)` que inverte uma árvore (sem criar uma nova árvore).
Define a function `void mirror (ABin *)` that inverts a tree (without creating a new tree).
(`https://codeboard.io/projects/16268/summary`)

31. Defina a função `void inorder (ABin , LInt *)` que cria uma lista ligada de inteiros a partir de uma travessia *inorder* de uma árvore binária.

Define the function ***void inorder (ABin , LInt *)*** *that creates a linked list of integers from an* inorder *traversal of a binary tree.*
(`https://codeboard.io/projects/16269/summary`)

32. Defina a função `void preorder (ABin , LInt *)` que cria uma lista ligada de inteiros a partir de uma travessia *preorder* de uma árvore binária.
Define the function ***void preorder (ABin , LInt *)*** *that creates a linked list of integers from a* preorder *traversal of a binary tree.*
(`https://codeboard.io/projects/16270/summary`)

33. Defina a função `void posorder (ABin , LInt *)` que cria uma lista ligada de inteiros a partir de uma travessia *posorder* de uma árvore binária.
Define the function ***void posorder (ABin , LInt *)*** *that creates a linked list of integers from a* posorder *traversal of a binary tree.*
(`https://codeboard.io/projects/16272/summary`)

34. Apresente uma definição da função `int depth (ABin a, int x)` que calcula o nível (menor) a que um elemento está numa árvore binária (`-1` caso não exista).
Define the function ***int depth (ABin a, int x)*** *which calculates the (lowest) level at which an element appears in a binary tree (`-1` if it doesn't exist).*
(`https://codeboard.io/projects/16273/summary`)

35. Defina uma função `int freeAB (ABin a)` que liberta o espaço ocupado por uma árvore binária, retornando o número de nodos libertados.
Define a function ***int freeAB (ABin a)*** *that frees the space used by a binary tree, returning the number of freed nodes.*
(`https://codeboard.io/projects/16274/summary`)

36. Defina uma função `int pruneAB (ABin *a, int l)` que remove (libertando o espaço respectivo) todos os elementos da árvore `*a` que estão a uma profundidade superior a `l`, retornando o número de elementos removidos.
Assuma que a profundidade da raíz da árvore é 1, e por isso a invocação `pruneAB(&a,0)` corresponde a remover todos os elementos da árvore `a`.
Define a function ***int pruneAB (ABin *a, int l)*** *that removes (by freeing the respective space) all the elements of the tree* ****a*** *that are deeper than* `l`*, returning the number of elements removed.*
Assume that the depth of the tree root is 1, so the invocation ***pruneAB(&a,0)*** *corresponds to removing all the elements from the tree* `a`*.*
(`https://codeboard.io/projects/16275/summary`)

37. Defina uma função `int iguaisAB (ABin a, ABin b)` que testa se duas árvores são iguais (têm os mesmos elementos e a mesma forma).
Define a function ***int iguaisAB(ABin a, ABin b)*** *that tests whether two trees are equal (have the same elements and the same shape).*
(`https://codeboard.io/projects/16276/summary`)

38. Defina uma função `LInt nivelL (ABin a, int n)` que, dada uma árvore binária, constrói uma lista com os valores dos elementos que estão armazenados na árvore ao nível `n` (assuma que a raiz da árvore está ao nível `1`).
Define a function ***LInt nivelL (ABin a, int n)*** *which, given a binary tree, builds a list with the values of the elements that are stored in the tree at level* ***n*** *(assume that the root of the tree is at level*

1).
(`https://codeboard.io/projects/16277/summary`)

39. Defina uma função `int nivelV (ABin a, int n, int v[])` que preenche o vector `v` com os elementos de `a` que se encontram no nível `n`.
Considere que a raíz da árvore se encontra no nível `1`.
A função deverá retornar o número de posições preenchidas do array.
Define a function `int nivelV (ABin a, int n, int v[])` that fills the vector `v` with the elements of `a` that are at the `n` level.
Consider that the root of the tree is at the `1` level.
The function should return the number of filled positions in the array.
(`https://codeboard.io/projects/16278/summary`)

40. Defina uma função `int dumpAbin (ABin a, int v[], int N)` que dada uma árvore `a`, preenche o array `v` com os elementos da árvore segundo uma travessia inorder. A função deverá preencher no máximo `N` elementos e retornar o número de elementos preenchidos.
Define a function `int dumpAbin (ABin a, int v[], int N)` that, given a tree `a`, fills the array `v` with the elements of the tree according to an inorder traversal. The function should fill at most `N` elements and return the number of elements filled.
(`https://codeboard.io/projects/16279/summary`)

41. Defina uma função `ABin somasAcA (ABin a)` que, dada uma árvore de inteiros, calcula a árvore das somas acumuladas dessa árvore.
A árvore calculada deve ter a mesma forma da árvore recebida como argumento e em cada nodo deve conter a soma dos elementos da sub-árvore que aí se inicia.
Define a function `ABin somasAcA (ABin a)` which, given a tree of integers, calculates the tree of accumulated sums of that tree.
The calculated tree must have the same shape as the tree received as an argument and at each node it must contain the sum of the elements of the subtree that starts there.
(`https://codeboard.io/projects/16280/summary`)

42. Apresente uma definição da função `int contaFolhas (ABin a)` que dada uma árvore binária de inteiros, conta quantos dos seus nodos são folhas, i.e., que não têm nenhum descendente.
Give a definition of the function `int contaFolhas (ABin a)` which, given a binary tree of integers, counts how many of its nodes are leaves, i.e. have no descendants.
(`https://codeboard.io/projects/16281/summary`)

43. Defina uma função `ABin cloneMirror (ABin a)` que cria uma árvore nova, com o resultado de inverter a árvore (efeito de espelho).
Define a function `ABin cloneMirror (ABin a)` that creates a new tree, with the result of inverting the tree (mirror effect).
(`https://codeboard.io/projects/16282/summary`)

44. Apresente uma definição não recursiva da função `int addOrd (ABin *a, int x)` que adiciona um elemento a uma árvore binária de procura. A função deverá retornar `1` se o elemento a inserir já existir na árvore ou `0` no outro caso.
Give a non-recursive definition of the function `int addOrd (ABin *a, int x)` which adds an element to a binary search tree. The function should return `1` if the element to be inserted already exists in the tree or `0` otherwise.
(`https://codeboard.io/projects/16283/summary`)

45. Apresente uma definição não recursiva da função `int lookupAB (ABin a, int x)` que testa se um elemento pertence a uma árvore binária de procura.

    Give a non-recursive definition of the function ***`int lookupAB (ABin a, int x)`*** *that tests whether an element belongs to a binary search tree.*

    (`https://codeboard.io/projects/16284/summary`)

46. Apresente uma definição da função`int depthOrd (ABin a, int x)` que calcula o nível a que um elemento está numa árvore binária de procura (`-1` caso não exista).

    Give a definition of the function ***`int depthOrd (ABin a, int x)`*** *which calculates the level at which an element is in a binary search tree (`-1` if it doesn't exist).*

    (`https://codeboard.io/projects/16285/summary`)

47. Apresente uma definição não recursiva da função `int maiorAB (ABin)` que calcula o maior elemento de uma árvore binária de procura não vazia.

    Give a non-recursive definition of the function ***`int maiorAB(ABin)`*** *which calculates the largest element of a non-empty binary search tree.*

    (`https://codeboard.io/projects/16286/summary`)

48. Defina uma função `void removeMaiorA (ABin *)` que remove o maior elemento de uma árvore binária de procura.

    Define a function ***`void removeMaiorA (ABin *)`*** *that removes the largest element from a binary search tree.*

    (`https://codeboard.io/projects/16287/summary`)

49. Apresente uma definição da função `int quantosMaiores (ABin a, int x)` que, dada uma árvore binária de procura de inteiros e um inteiro, conta quantos elementos da árvore são maiores que o inteiro dado.

    Give a definition of the function ***`int quantosMaiores (ABin a, int x)`*** *which, given a binary integer search tree and an integer, counts how many elements of the tree are greater than the given integer.*

    (`https://codeboard.io/projects/16288/summary`)

50. Apresente uma definição da função `void listToBTree (LInt l, ABin *a)` que constrói uma árvore binária de procura a partir de uma lista ligada ordenada.

    Give a definition of the function ***`void listToBTree (LInt l, ABin *a)`*** *which builds a binary search tree from an ordered linked list.*

    (`https://codeboard.io/projects/16289/summary`)

51. Apresente uma definição da função `int deProcura (ABin a)` que testa se uma árvore é de procura.

    Give a definition of the function ***`int deProcura (ABin a)`*** *which tests whether a tree is a binary search tree.*

    (`https://codeboard.io/projects/16290/summary`)