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;
```

- Apresente uma definição não recursiva da função int length (LInt) que calcula o comprimento de uma lista ligada. (https://codeboard.io/projects/16161)
- Apresente uma definição não recursiva da função void freeL (LInt) que liberta o espaço ocupado por uma lista.
- 3. Apresente uma definição não recursiva da função void imprimeL (LInt) que imprime no ecran os elementos de uma lista (um por linha).
- 4. Apresente uma definição não recursiva da função LInt reverseL (LInt) que inverte uma lista (sem criar uma nova lista). (https://codeboard.io/projects/16243)
- 5. Apresente uma definição não recursiva da função void insertOrd (LInt *, int) que insere ordenadamente um elemento numa lista ordenada. (https://codeboard.io/projects/16244)
- 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. (https://codeboard.io/projects/16245)
- 7. Defina uma função merge (LInt *r, LInt a, LInt b) que junta duas listas ordenadas (a e b) numa única lista ordenada (r). (https://codeboard.io/projects/16246)
- 8. Defina uma função void splitQS (LInt 1, int x, LInt *mx, LInt *Mx) que, dada uma lista ligada 1 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 1 menores do que x e a outra com os restantes. Note que esta função não deverá criar cópias dos elementos da lista. (https://codeboard.io/projects/16247)
- 9. Defina uma função LLig parteAmeio (LLig *1) que, parte uma lista não vazia *1 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} (https://codeboard.io/projects/17392)

- 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. (https://codeboard.io/projects/16249)
- 11. Apresente uma definição da função int removeDups (LInt *) que remove os valores repetidos de uma lista (deixando apenas a primeira ocorrência). (https://codeboard.io/projects/16250)
- 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. (https://codeboard.io/projects/16251)
- 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). (https://codeboard.io/projects/16252)
- 14. Apresente uma definição não recursiva da função void appendL (LInt *, int) que acrescenta um elemento no fim da lista. (https://codeboard.io/projects/16253)
- 15. Apresente uma definição da função void concatL (LInt *a, LInt b) que acrescenta a lista b à lista *a. (https://codeboard.io/projects/16254)
- 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.
- 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 1 tiver 5 elementos com os valores {1,2,3,4,5} por esta ordem, a invocação cloneRev(1) deve corresponder a uma nova lista com os elementos {5,4,3,2,1} por esta ordem. (https://codeboard.io/projects/16256)
- 18. Defina uma função int maximo (LInt 1) que calcula qual o maior valor armazenado numa lista não vazia. (https://codeboard.io/projects/16257)
- 19. Apresente uma definição iterativa da função int take (int n, LInt *1) que, dado um inteiro n e uma lista ligada de inteiros 1, apaga de 1 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. (https://codeboard.io/projects/16258)
- 20. Apresente uma definição iterativa da função int drop (int n, LInt *1) que, dado um inteiro n e uma lista ligada de inteiros 1, apaga de 1 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. (https://codeboard.io/projects/16259)
- 21. O tipo LInt pode ser usado ainda para implementar listas circulares. Defina uma função LInt Nforward (LInt 1, int N) que, dada uma lista circular dá como resultado o endereço do elemento da lista que está N posições à frente. (https://codeboard.io/projects/16260)

- 22. Defina uma função int listToArray (LInt 1, int v[], int N) que, dada uma lista 1, preenche o array v com os elementos da lista.
 - A função deverá preencher no máximo N elementos e retornar o número de elementos preenchidos. (https://codeboard.io/projects/16261)
- 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.. (https://codeboard.io/projects/16262)
- 24. Defina uma função LInt somasAcL (LInt 1) 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 1 tiver os valores [1,2,3,4] a lista contruída pela invocação de somasAcL (1) deverá conter os valores [1,3,6,10]. (https://codeboard.io/projects/16263)
- 25. Defina uma função void remreps (LInt 1) 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. (https://codeboard.io/projects/16264)
- 26. Defina uma função LInt rotatel (LInt 1) 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. (https://codeboard.io/projects/16265)
- 27. Defina uma função LInt parte (LInt 1) que parte uma lista 1 em duas: na lista 1 ficam apenas os elementos das posições ímpares; na lista resultante ficam os restantes elementos.

Assim, se a lista x tiver os elementos $\{1,2,3,4,5,6\}$ a chamada y = parte (x), coloca na lista y os elementos $\{2,4,6\}$ ficando em x apenas os elementos $\{1,3,5\}$ (https://codeboard.io/projects/16266)

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. (https://codeboard.io/projects/16220)
- 29. Defina uma função ABin cloneAB (ABin) que cria uma cópia de uma árvore. (https://codeboard.io/projects/16267)
- 30. Defina uma função void mirror (ABin *) que inverte uma árvore (sem criar uma nova árvore). (https://codeboard.io/projects/16268)

- 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. (https://codeboard.io/projects/16269)
- 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. (https://codeboard.io/projects/16270)
- 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. (https://codeboard.io/projects/16272)
- 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). (https://codeboard.io/projects/16273)
- 35. Defina uma função int freeAB (ABin a) que liberta o espaço ocupado por uma árvore binária, retornando o número de nodos libertados. (https://codeboard.io/projects/16274)
- 36. Defina uma função int pruneAB (ABin *a, int 1) que remove (libertando o espaço respectivo) todos os elementos da árvore *a que estão a uma profundidade superior a 1, 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. (https://codeboard.io/projects/16275)
- 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). (https://codeboard.io/projects/16276)
- 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). (https://codeboard.io/projects/16277)
- 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. (https://codeboard.io/projects/16278)
- 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. (https://codeboard.io/projects/16279)
- 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. (https://codeboard.io/projects/16280)
- 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. (https://codeboard.io/projects/16281)

- 43. Defina uma função ABin cloneMirror (ABin a) que cria uma árvore nova, com o resultado de inverter a árvore (efeito de espelho). (https://codeboard.io/projects/16282)
- 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. (https://codeboard.io/projects/16283)
- 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. (https://codeboard.io/projects/16284)
- 46. Apresente uma definição da funçãoint depthOrd (ABin a, int x) que calcula o nível a que um elemento está numa árvore binária de procura (-1 caso não exista). (https://codeboard.io/projects/16285)
- 47. Apresente uma definição não recursiva da função int maiorAB (ABin) que calcula o maior elemento de uma árvore binária de procura não vazia. (https://codeboard.io/projects/16286)
- 48. Defina uma função void removeMaiorA (ABin *) que remove o maior elemento de uma árvore binária de procura. (https://codeboard.io/projects/16287)
- 49. Apresente uma definição da função int quantos Maiores (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. (https://codeboard.io/projects/16288)
- 50. Apresente uma definição da função void listToBTree (LInt 1, ABin *a) que constrói uma árvore binária de procura a partir de uma lista ligada ordenada. (https://codeboard.io/projects/16289)
- 51. Apresente uma definição da função int deProcura (ABin a) que testa se uma árvore é de procura. (https://codeboard.io/projects/16290)