Métodos de Programação II

MIETI

Exercícios sobre Programação em C

11 Fevereiro de 2021

Capítulo 1

Exercícios com Tipos de Dados Nativos e Estruturados Estáticos

1.1 Inteiros

1. Defina um programa que lê (usando a função `scanf` uma sequência de números inteiros terminada com o número 0 e imprime no ecrã a soma dos números lidos.

2. Defina um programa que lê (usando a função `scanf` uma sequência de números inteiros terminada com o número 0 e imprime no ecrã o maior elemento da sequência.

3. Defina um programa que lê (usando a função `scanf` uma sequência de números inteiros terminada com o número 0 e imprime no ecrã a média da sequência.

4. Defina um programa que lê (usando a função `scanf` uma sequência de números inteiros terminada com o número 0 e imprime no ecrã o segundo maior elemento.

5. Defina uma função `int bitsUm (unsigned int n)` que calcula o número de bits iguais a 1 usados na representação binária de um dado número n.

   `https://codeboard.io/projects/13548`

6. Defina uma função `int trailingZ (unsigned int n)` que calcula o número de bits a 0 no final da representação binária de um número (i.e., o expoente da maior potência de 2 que é divisor desse número).

   `https://codeboard.io/projects/13549`

7. Defina uma função `int qDig (unsigned int n)` que calcula o número de dígitos necessários para escrever o inteiro n em base decimal. Por exemplo, `qDig (440)` deve devolver 3.

   `https://codeboard.io/projects/13583`

1.2 Strings

8. Apresente uma definição da função pré-definida em C `int strlen (char str[])` que calcula o comprimento de uma string.

   `https://codeboard.io/projects/13584`

9. Apresente uma definição da função pré-definida em C `char *strcat (char s1[], char s2[])` que concatena a string $s2$ a $s1$ (devolvendo o endereço da primeira).

   `https://codeboard.io/projects/14490`

10. Apresente uma definição da função pré-definida em C `char *strcpy (char *dest, char source[])` que copia a string *source* para *dest* devolvendo o valor desta última.

    `https://codeboard.io/projects/14491`

11. Apresente uma definição da função pré-definida em C `int strcmp (char s1[], char s2[])` que compara (lexicograficamente) duas strings. O resultado deverá ser:

    - 0 se as strings forem iguais
    - $<$0 se s1 $<$ s2
    - $>$0 se s1 $>$ s2

    `https://codeboard.io/projects/14492`

12. Apresente uma definição da função pré-definida em C `char *strstr (char s1[], char s2[])` que determina a posição onde a string $s2$ ocorre em $s1$. A função devolve um apontador para a posição em que $s2$ ocorre em $s1$, ou `NULL` caso $s2$ não ocorra em $s1$.

    `https://codeboard.io/projects/14493`

13. Defina uma função `void strrev (char s[])` que inverte uma string.

    `https://codeboard.io/projects/14494`

14. Defina uma função `void strnoV (char s[])` que retira todas as vogais de uma string.

    `https://codeboard.io/projects/13661`

15. Defina uma função `void truncW (char t[], int n)` que dado um texto t com várias palavras (as palavras estão separadas em t por um ou mais espaços) e um inteiro n, trunca todas as palavras de forma a terem no máximo n carateres. Por exemplo, se a string *txt* contiver "liberdade, igualdade e fraternidade", a invocação de *truncW (txt, 4)* deve fazer com que passe a estar lá armazenada a string "libe igua e frat".

    `https://codeboard.io/projects/13659`

16. Defina uma função `char charMaisfreq (char s[])` que determina qual o carater mais frequente numa string. A função deverá devolver 0 no caso de s ser a string vazia.

    `https://codeboard.io/projects/14577`

17. Defina uma função `int iguaisConsecutivos (char s[])` que, dada uma string s calcula o comprimento da maior sub-string com carateres iguais. A função deve devolver 0 se a string for vazia e devolver 1 quando a string não é vazia mas não há qualquer sequência com carateres iguais. Outro exemplo, `iguaisConsecutivos ("aabcccaac")` deve dar como resultado 3, correspondendo à repetição "ccc".

    `https://codeboard.io/projects/14578`

18. Defina uma função `int difConsecutivos (char s[])` que, dada uma string s calcula o comprimento da maior sub-string com carateres diferentes. Por exemplo, `difConsecutivos ("aabcccaac")` deve dar como resultado 3, correspondendo à string "abc".

    `https://codeboard.io/projects/14579`

19. Defina uma função `int maiorPrefixo (char s1 [], char s2 [])` que calcula o comprimento do maior prefixo comum entre as duas strings.

    `https://codeboard.io/projects/14580`

20. Defina uma função `int maiorSufixo (char s1 [], char s2 [])` que calcula o comprimento do maior sufixo comum entre as duas strings.

    `https://codeboard.io/projects/14581`

21. Defina a função `int sufPref (char s1[], char s2[])` que calcula o tamanho do maior sufixo de $s1$ que é um prefixo de $s2$. Por exemplo, `sufPref("batota","totalidade")` deve dar como resultado 4, uma vez que a string "tota"é um sufixo de "batota"e um prefixo de "totalidade".

    `https://codeboard.io/projects/14582`

22. Defina uma função `int contaPal (char s[])` que conta as palavras de uma string. Uma palavra é uma sequência de carateres (diferentes de espaço) terminada por um ou mais espaços. Assim se a string p tiver o valor "a a bb a", o resultado de `contaPal (p)` deve ser 4.

    `https://codeboard.io/projects/14583`

23. Defina uma função `int contaVogais (char s[])` que devolve o número de vogais da string s. Não se esqueça de considerar tanto maiúsculas como minúsculas.

    `https://codeboard.io/projects/14585`

24. Defina uma função `int contida (char a[], char b[])` que testa se todos os carateres da primeira string também aparecem na segunda. Por exemplo, `contida ("braga", "bracara augusta")` deve devolver *verdadeiro* enquanto que `contida ("braga", "bracarense")` deve devolver *falso*.

    `https://codeboard.io/projects/14586`

25. Defina uma função `int palindrome (char s[])` que testa se uma palavra é palíndrome, i.e., lê-se de igual forma nos dois sentidos.

    `https://codeboard.io/projects/14587`

26. Defina uma função `int remRep (char x[])` que elimina de uma string todos os carateres que se repetem sucessivamente deixando lá apenas uma cópia. A função deverá devolver o comprimento da string resultante. Assim, por exemplo, ao invocarmos a função com um vetor contendo "aaabaaabbbaaa", o vetor deve passar a conter a string "ababa"e a função deverá devolver o valor 5.

    `https://codeboard.io/projects/13663`

27. Defina uma função `int limpaEspacos (char t[])` que elimina repetições sucessivas de espaços por um único espaço. A função deve devolver o comprimento da string resultante.

    `https://codeboard.io/projects/13733`

1.3 Arrays (ou Vetores)

28. Defina uma função `void insere (int v[], int N, int x)` que insere um elemento x num vetor v. Assuma que o vetor v tem um tamanho maior do que N, que as posições que estão realmente ocupadas são as N primeiras e que estão ordenadas por ordem crescente dos valores guardados. Por isso, após a inserção de x, o vetor deverá ter as primeiras $N+1$ posições ordenadas por ordem crescente.

    https://codeboard.io/projects/14836

29. Defina uma função `void merge (int r [], int a[], int b[], int na, int nb)` que, dados vetores ordenados a (com na elementos) e b (com nb elementos), preenche o vetor r (com na+nb elementos) com os elementos de a e b ordenados.

    https://codeboard.io/projects/14837

30. Defina uma função `int crescente (int a[], int i, int j)` que testa se os elementos do vetor a, entre as posições i e j (inclusive) estão ordenados por ordem crescente. A função deve devolver 1 ou 0 consoante o vetor esteja ou não ordenado.

    https://codeboard.io/projects/14838

31. Defina uma função `int retiraNeg (int v[], int N)` que retira os números negativos de um vetor com N inteiros. A função deve devolver o número de elementos que não foram retirados.

    https://codeboard.io/projects/14839

32. Defina uma função `int menosFreq (int v[], int N)` que recebe um vetor v com N elementos **ordenado por ordem crescente** e devolve o **menos frequente** dos elementos do vetor. Se houver mais do que um elemento nessas condições deve devolver o que aparece no índice mais baixo.

https://codeboard.io/projects/14840

33. Defina uma função `int maisFreq (int v[], int N)` que recebe um vetor v com N elementos **ordenado por ordem crescente** e devolve o **mais frequente** dos elementos do vetor. Se houver mais do que um elemento nessas condições deve devolver o que aparece no índice mais baixo.

    https://codeboard.io/projects/14841

34. Defina uma função `int maxCresc (int v[], int N)` que calcula o comprimento da maior sequência crescente de elementos consecutivos num vetor v com N elementos. Por exemplo, se o vetor contiver 10 elementos pela seguinte ordem: `1, 2, 3, 2, 1, 4, 10, 12, 5, 4`, a função deverá devolver 4, correspondendo ao tamanho da sequência `1, 4, 10, 12`.

    https://codeboard.io/projects/14842

35. Defina uma função `int elimRep (int v[], int n)` que recebe um vetor v com n inteiros e elimina as repetições. A função deverá devolver o número de elementos do vetor resultante. Por exemplo, se o vetor v contiver nas suas primeiras 10 posições os números `{1, 2, 3, 2, 1, 4, 2, 4, 5, 4}` a invocação `elimRep (v,10)` deverá devolver 5 e colocar nas primeiras 5 posições do vetor os elementos `{1,2,3,4,5}`.

    https://codeboard.io/projects/14843

36. Defina uma função `int elimRepOrd (int v[], int n)` que recebe um vetor v com n inteiros ordenado por ordem crescente e elimina as repetições. A função deverá devolver o número de elementos do vetor resultante.

    https://codeboard.io/projects/14844

37. Defina uma função `int comunsOrd (int a[], int na, int b[], int nb)` que calcula quantos elementos os vetores a (com na elementos) e b (com nb elementos) têm em comum. Assuma que os vetores a e b estão ordenados por ordem crescente.

    https://codeboard.io/projects/14845

38. Defina uma função `int comuns (int a[], int na, int b[], int nb)` que calcula quantos elementos os vetores a (com na elementos) e b (com nb elementos) têm em comum. Assuma que os vetores a e b não estão ordenados e não têm elementos repetidos. Defina a função sem alterar os vetores recebidos como argumentos.

    https://codeboard.io/projects/14846

39. Defina uma função `int minInd (int v[], int n)` que, dado um vetor v com n inteiros, devolve o índice do menor elemento do vetor.

    https://codeboard.io/projects/14847

40. Defina uma função `void somasAc (int v[], int Ac [], int N)` que preenche o vetor Ac com as somas acumuladas do vetor v. Por exemplo, o conteúdo da posição `Ac[3]` deve ser calculado como `v[0]+v[1]+v[2]+v[3]`.

    `https://codeboard.io/projects/14848`

1.4 Arrays 2D (ou Matrizes)

41. Defina uma função `int triSup (int N, int m [N][N])` que testa se uma matriz quadrada é triangular superior, i.e., se todos os elementos abaixo da diagonal são zeros. A função devolve 1 se a matriz for triangular superior, caso contrário devolve 0.

    https://codeboard.io/projects/14849

42. Defina uma função `void transposta (int N, float m [N][N])` que transforma uma matriz quadrada na sua transposta.

    https://codeboard.io/projects/14850

43. Defina uma função `void addTo (int N, int M, int a [N][M], int b[N][M])` que adiciona a segunda matriz à primeira.

    https://codeboard.io/projects/14851

44. Escreva em C uma função `void sumDiag (int N, int m [N][N])` que recebe como argumento uma matriz quadrada e substitui cada elemento da diagonal pelo somatório de todos os outros elementos dessa **linha**. O exemplo abaixo mostra o efeito desta função numa matriz 4×4.

$$\left|\begin{array}{cccc} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{array}\right| \quad \Longrightarrow \quad \left|\begin{array}{cccc} 9 & 2 & 3 & 4 \\ 5 & 20 & 7 & 8 \\ 9 & 10 & 31 & 12 \\ 13 & 14 & 15 & 42 \end{array}\right|$$

    https://codeboard.io/projects/14852

45. Escreva um programa que liste no ecrã as letras do alfabeto (maiúsculas e minúsculas) e o respectivo código ASCII. Use para isso a função `printf`, tanto para imprimir os carateres como os seus códigos (inteiros).

1.5 Conjuntos e multi-conjuntos

46. Uma forma de representar conjuntos de índices consiste em usar um array de inteiros contendo 1 ou 0 consoante esse índice pertence ou não ao conjunto. Assim o conjunto `{1, 4, 7}` seria representado por um array em que as primeiras 8 posições conteriam `{0,1,0,0,1,0,0,1}`. Apresente uma definição da função `int unionSet (int N, int v1[N], int v2[N], int r[N])` que coloca no array r o resultado da união dos conjuntos $v1$ e $v2$. Como se pode deduzir da assinatura da função, assuma que os 3 arrays ($v1$, $v2$ e r) têm N elementos e que a função devolve o número de índices contidos na união de $v1$ e $v2$.

    https://codeboard.io/projects/14685

47. Uma forma de representar conjuntos de índices consiste em usar um array de inteiros contendo 1 ou 0 consoante esse índice pertença ou não ao conjunto. Assim o conjunto `{1, 4, 7}` seria representado por um array em que as

primeiras 8 posições conteriam `{0,1,0,0,1,0,0,1}`. Apresente uma definição da função `int intersectSet (int N, int v1[N], int v2[N], int r[N])` que coloca no array r o resultado da interseção dos conjuntos $v1$ e $v2$. Como se pode deduzir da assinatura da função, assuma que os 3 arrays ($v1$, $v2$ e r) têm N elementos e que a função devolve o número de índices contidos na interseção de $v1$ e $v2$.

https://codeboard.io/projects/14694

48. Uma forma de representar multi-conjuntos de índices consiste em usar um array de inteiros contendo em cada posição o número de ocorrências desse índice. Assim o multi-conjunto `{1, 1, 4, 7, 7, 7}` seria representado por um array em que as primeiras 8 posições conteriam `{0,2,0,0,1,0,0,3}`. Apresente uma definição da função `int intersectMSet (int N, int v1[N], int v2[N], int r[N])` que coloca no array r o resultado da intersecção dos multi-conjuntos $v1$ e $v2$. Como se pode deduzir da assinatura da função, assuma que os 3 arrays ($v1$, $v2$ e r) têm N elementos e que a função devolve o total de ocorrências contidas na interseção de $v1$ e $v2$.

    https://codeboard.io/projects/14733

49. Uma forma de representar multi-conjuntos de índices consiste em usar um array de inteiros contendo em cada posição o número de ocorrências desse índice. Assim o multi-conjunto `{1, 1, 4, 7, 7, 7}` seria representado por um array em que as primeiras 8 posições conteriam `{0,2,0,0,1,0,0,3}`. Apresente uma definição da função `int unionMSet (int N, int v1[N], int v2[N], int r[N])` que coloca no array r o resultado da união dos multi-conjuntos $v1$ e $v2$. Como se pode deduzir da assinatura da função, assuma que os 3 arrays ($v1$, $v2$ e r) têm N elementos e que a função devolve o número de índices contidos na união de $v1$ e $v2$.

    https://codeboard.io/projects/14734

50. Uma forma de representar multi-conjuntos de índices consiste em usar um array de inteiros contendo em cada posição o número de ocorrências desse índice. Assim o multi-conjunto `{1, 1, 4, 7, 7, 7}` seria representado por um array em que as primeiras 8 posições conteriam `{0,2,0,0,1,0,0,3}`. Apresente uma definição da função `int cardinalMSet (int N, int v[N])` que calcula a número de elementos do multi-conjunto v.

    https://codeboard.io/projects/14740

Capítulo 2

Exercícios com Tipos de Dados Estruturados Heterogéneos e Dinâmicos

2.1 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.

   https://codeboard.io/projects/16161

2. Apresente uma definição não recursiva da função `void freeL (LInt)` que liberta o espaço ocupado por uma lista ligada.

   https://codeboard.io/projects/79669

3. Apresente uma definição não recursiva da função `void imprimeL (LInt)` que imprime no ecrã os elementos de uma lista ligada (um por linha).

   https://codeboard.io/projects/79670

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 splitMS (LInt l, int x, LInt *mx, LInt *Mx)` que, dada uma lista ligada l e um inteiro x, parte a lista em duas (devolvendo o endereço do primeiro elemento de cada lista em $*mx$ e $*M$x): 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.

   https://codeboard.io/projects/16247

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}`.

   https://codeboard.io/projects/17392

10. Apresente uma definição não recursiva da função `int removeAll (LInt *, int n)` que remove todas as ocorrências do inteiro n de uma lista, devolvendo 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 d') o maior elemento de uma lista não vazia, devolvendo 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 v)` que acrescenta um elemento v 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 l)` que cria uma nova lista ligada com os elementos pela ordem em que aparecem na lista l passada como 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 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.

    https://codeboard.io/projects/16256

18. Defina uma função `int maximo (LInt l)` 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 *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 devolver o comprimento final da lista.

    https://codeboard.io/projects/16258

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 devolver 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 l, 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 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 devolver 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 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 construída pela invocação de `somasAcL (l)` deverá conter os valores `[1,3,6,10]`.

https://codeboard.io/projects/16263

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 nodos removidos é correctamente libertado.

    https://codeboard.io/projects/16264

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.

    https://codeboard.io/projects/16265

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 `{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

2.2 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 (profundidade) 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 `LInt * 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, devolvendo o número de nodos libertados.

    https://codeboard.io/projects/16274

36. Defina uma função `int pruneAB (ABin *a, int l)` que remove (libertando o espaço respetivo) todos os elementos da árvore $*a$ que estão a uma profundidade superior a l, devolvendo 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 raíz 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 vetor 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 devolver 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 devolver 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

2.3 Árvores Binárias de Pesquisa

44. Apresente uma definição não recursiva da função `int addOrd (ABin *a, int x)` que adiciona um elemento x a uma árvore binária de pesquisa a. A função deverá devolver 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 x pertence a uma árvore binária de pesquisa a. A função deverá devolver 1 se o elemento x existir na árvore ou 0 no outro caso.

    https://codeboard.io/projects/16284

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

    https://codeboard.io/projects/16286

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

    https://codeboard.io/projects/16287

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

    https://codeboard.io/projects/16288

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

    https://codeboard.io/projects/16289

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

    https://codeboard.io/projects/16290