

Esta prova tem a duração de **2 horas** e é **sem consulta**.

1. Implemente o método *desfolhar*(  $AB<T> arv$  ) que retorne a árvore binária que se obtém da árvore binária passada por parâmetro, *arv*, por remoção de todas as suas folhas. Apresente a complexidade do algoritmo implementado.
2. Elabore o método *getLE*(  $T x, ABP<T> arv$  ) que retorne uma lista com os nós da árvore binária de pesquisa *arv*, que são menores ou iguais a  $x$ . Implemente um algoritmo o mais eficiente possível e indique a sua complexidade, relacionando-a com  $n'$ , o número de nós menores ou iguais a  $x$  e  $h$  a altura de árvore.
3. Se sobre uma árvore binária de pesquisa, realizar a operação *insere*( $x$ ) seguida de *remove*( $x$ ), para exactamente o mesmo valor  $x$ , obtemos, em qualquer circunstância, a árvore inicial? Justifique a sua resposta.
4. Considere a seguinte sequência de valores, 7;2;1;14;35;19;20;54;18;43;40;34;42.
  - (a) Apresente as árvores AVL, resultantes de inserir numa árvore inicialmente vazia, a sequência de valores dada, identificando em caso de desequilíbrio, o nó correspondente, o caso da rotação e se a resolução passa por uma rotação simples ou dupla
  - (b) Apresente o resultado do percurso post-ordem da árvore obtida anteriormente
5. Desenhe numa tabela de hash de tamanho 11 ( $N = 11$ ), os resultados de inserir as chaves: 6; 4; 18; 3; 33; 9; 17; 5; 29; 101 tomando para função de hash,  $f(x) = (2 * (x \bmod N)) + 1$  e usando:
  - (a) Hasinhg fechado com acesso quadrático
  - (b) Hashing fechado com duplo hashing, usando como segunda função de hash,  $g(x) = 5 - (x \bmod 5)$
6. Considere o array da figura 1. Ordene-o usando os seguintes métodos:
  - (a) *Mergesort*
  - (b) *Heapsort*: apresente todas as iterações do método, e as respectivas heaps
  - (c) Uma única iteração do *Quicksort*, até à obtenção da primeira partição e usando como pivot o elemento na posição índice 5, indique claramente qual a partição esquerda e direita bem como os elementos em cada posição do array.

0	1	2	3	4	5	6	7	8	9	10
4	12	1	33	56	20	17	71	6	19	15

Figure 1: array de valores