DEEC
DEPARTAMENTO DE ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES
TÉCNICO LISBOA

AED - 2015-2016 - 2º Semestre
Algoritmos e Estruturas de Dados

1º Exame, 4 Junho 2016, 08:00h Duração: 3 horas

Prova escrita, individual e sem consulta

NOME:_____ NÚMERO:_____

PARTE I - Questões de Escolha Múltipla (A/B/C/D)

Preencha as respostas na tabela (usando apenas letras maiúsculas). Se nenhuma opção servir, escreva **NENHUMA**. Se pretender alterar a sua resposta, risque e escreva ao lado a sua nova opção. Todas as questões de escolha múltipla seguintes valem 0.75 valores. Estas questões de escolha múltipla não respondidas são cotadas com 0 valores, mas por cada resposta errada são descontados 0.75/4 valores.

Questão	1	2	3	4	5	6	7	8
Resposta								

[0.75] 1. Considere o código à esquerda e indique para que valor de `n` são realizadas mais chamadas recursivas

```
void superWriteVertically(int n) {
 if (n < 0) {
  printf("-\n");
  superWriteVertically(-1*n);
 }
 else if (n < 10)
  printf("%d\n", n);
 else {
  superWriteVertically(n/10);
  printf("%d\n", n % 10);
 }
}
```

A. 0
B. 321
C. -34
D. -321

[0.75] 2. Para uma árvore binária, AVL ou não, qual das seguintes afirmações é condição suficiente para garantir ser ordenada?

A. O varrimento BFS produz uma sequência ordenada.
B. Para todos os vértices, a raíz da sub-árvore esquerda é menor e a raíz da sub-árvore direita é maior.
C. Para todos os vértices, todos os elementos na sub-árvore esquerda são menores e todos os elementos na sub-árvore direita são maiores.
D. A raíz da sub-árvore esquerda é menor e a raíz da sub-árvore direita é maior.

[0.75] 3. Considere o código abaixo à esquerda, analise-o e indique qual das recorrências propostas descreve a complexidade temporal da função apresentada.

```
int mystery(int n) {
 int answer;
 if (n > 0) {
  answer = (2*mystery(n-3) + 3*mystery(n/2) + 5);
  return answer;
 }
 else
  return 1;
}
```

A. $C_n = 2C_{n-3} + 3C_{n/2} + 5$
B. $C_n = C_{n-3} + C_{n/2} + 5$
C. $C_n = C_{n-3} + 3C_{n/2} + 5$
D. $C_n = 2C_{n-3} + C_{n/2} + 5$

[0.75] 4. Suponha que pretende resolver um problema de balanceamento de parêntesis, e para isso decide utilizar uma pilha (inicialmente vazia) e o pseudo-código abaixo.

```
declare a character stack

while ( more input is available )
{
     read a character
     if ( the character is not a '(' nor a ')' )
          continue
     if ( the character is a '(' )
          push it on the stack
     else if ( the character is a ')' and the stack is not empty )
          pop a character off the stack
     else
          print "unbalanced" and exit
}

print "balanced"
```

Para as duas seguintes strings S e T: $S \equiv$ "$(ola(aed(boa)sorte)$", $T \equiv$ "$((o))adeus)(final)$" qual será a resposta do código acima?

A.	S balanced e T unbalanced.	B.	S unbalanced e T unbalanced.
C.	S unbalanced e T balanced.	D.	S balanced e T balanced.

[0.75] 5. Considere a seguinte tabela de números inteiros.

4	6	2	0	7	1	9	8	3	5

Supondo que se aplica o algoritmo "quicksort" para ordenar a tabela em formato decrescente, qual das tabelas abaixo se obtém após o processo de partição utilizando como pivot o número 5?

A.

8	6	9	7	0	1	2	4	3	5

B.

8	6	9	7	5	1	2	4	3	0

C.

7	6	9	8	5	4	2	0	1	3

D.

6	7	9	8	5	4	2	0	1	3

[0.75] 6. Considere as tabelas apresentadas abaixo. Qual dessas tabelas pode ser obtida por aplicação do algoritmo de união rápida equilibrada?

A.

1	2	3	0	1	1	1	4	4	5

B.

9	0	0	0	0	0	9	9	9	9

C.

1	2	3	4	5	6	6	7	8	9

D.

0	0	0	0	0	1	1	1	6	2

[0.75] 7. Considere o acervo representado pela tabela abaixo em que a prioridade é por ordem alfabética inversa e o símbolo ? representa uma letra desconhecida.

Z	Y	R	L	U	N	K	A	K	H	G	?

Supondo que ao acervo acima se aplica uma operação de RemoveMax e se obtém o acervo abaixo

Y	U	R	L	?	N	K	A	K	H	G

indique qual o conjunto de letras em que todas podem ser a letra desconhecida.

A. $\{A, \ldots, Y\}$	B. $\{A, \ldots, U\}$
C. $\{A, \ldots, G\} \cup \{O, \ldots, U\}$	D. $\{H, \ldots, N\}$

[0.75] 8. Suponha que se pretende construir uma árvore binária ordenada, satisfazendo o balaneamento AVL, de tal forma que a árvore deverá permanecer balanceada após cada inserção. Contando cada rotação dupla como duas rotações, para uma árvore inicialmente vazia, indique quantas rotações são realizada se forem inseridos os seguintes elementos: 32; 70; 57; 43; 37; 85; 91.

A. 3	B. 4	C. 5	D. 6

PARTE II - Questões de Escolha Múltipla (V/F)

Preencha as respostas na tabela (usando apenas letras maiúsculas – **V**(erdadeira) ou **F**(alsa)). Todas as questões de escolha múltipla seguintes valem 0.50 valores. Estas questões de escolha múltipla não respondidas ou erradas são cotadas com 0 valores.

Questão	9	10	11	12	13	14	15	16
Resposta								

[0.50] 9. Para tabelas de grandes dimensões o algoritmo de ordenação "quicksort" é sempre mais rápido que o algoritmo "insertsort".

[0.50] 10. $N \log N + N/5 \notin \mathcal{O}(N^2)$.

[0.50] 11. Contando cada rotação dupla como duas rotações, para uma árvore AVL com $2^k - 1$ vértices, a inserção do 2^k-ésimo vértice pode obrigar a que se tenham de realizar 3 ou mais rotações para repor a propriedade de balanceamento.

[0.50] 12. Em tabelas de dispersão, a resolução de colisões por dupla dispersão é melhor que procura linear porque a primeira produz clusters de menor dimensão.

[0.50] 13. Dado um grafo $G = (V, E)$, se um vértice $v \in V$ for visitado no nível k por procura em largura a partir de algum vértice $s \in V$, então todos os caminhos de s para v possuem, no máximo, comprimento k.

[0.50] 14. Se dois algoritmos diferentes para o mesmo problema resolvem uma mesma instância desse problema com tempos de execução diferentes, então possuem complexidades computacionais necessariamente diferentes.

[0.50] 15. Seja G um grafo não direccionado e seja T a sua árvore mínima de suporte. Se todos os pesos das arestas de G forem incrementados de uma constante $c > 0$, então T continua sendo uma árvore mínima de suporte.

[0.50] 16. Grafos ponderados não direccionados podem admitir mais que uma árvore mínima de suporte.

PARTE III - Questões de Desenvolvimento

Responda a cada uma das questões de desenvolvimento em **folhas de exame separadas** e devidamente identificadas com nome e número.

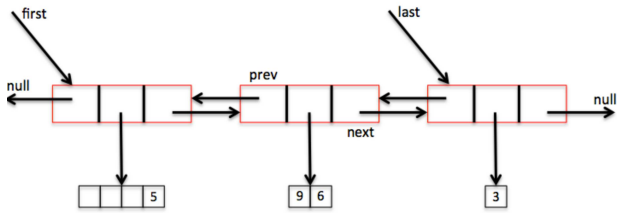
[4.0]

17. Neste problema considera-se uma implementação específica de uma estrutura de dados designada por Vector Dinâmico.

Nos vectores dinâmicos, quando o vector em uso está cheio e é preciso alojar mais um elemento, cria-se um vector com o dobro do tamanho para onde se copiam os elementos guardados no vector antigo, o qual é depois descartado.

Neste caso, considera-se uma estrutura de dados que explora uma variante disto: quando já não há espaço livre no vector em uso, cria-se um vector com o dobro do tamanho mas não se copia nem se descarta o vector antigo. Em vez disso, o novo vector irá servir para guardar apenas os elementos que venham a ser adicionados. Os vários vetores que vão sendo criados são guardados numa lista duplamente ligada por ordem inversa à ordem da criação (ou seja, o vector mais recente é o que está logo no início).

Na figura seguinte é apresentada uma representação gráfica do vector dinâmico $[3, 6, 9, 5]$.



Como estrutura de dados para este problema considere o seguinte código:

```
struct vecDnode {
        int * array;
        struct vecDnode * next;
        struct vecDnode * prev;
        int num;
        int dim;
};
```

Quando o vector dinâmico está vazio e pretende-se inserir o número 3, é criado um nó da lista e alocado um vector de inteiros de dimensão 1. A seguir, quando se insere o número 6, verifica-se que o vector anterior está cheio (dim = 1 com um inteiro inserido, ou seja, num = 1), é necessário criar um novo nó com um vetor de inteiros de dimensão 2. Este nó é inserido no início da lista e o número 6 é colocado na posição mais à direita do vector de inteiros. Quando se insere o 9, ainda existe espaço no primeiro nó (dim = 2 e num = 1), o 9 é colocado na posição livre mais à direita do vector de inteiros. Quando se insere o 5, verifica-se que o vector de inteiros do primeiro nó está cheio (dim = 2 e num = 2), é necessário criar um novo nó com um vector de inteiros de dimensão 4. O nó é inserido no início da lista e o 5 é colocado na posição mais à direita do vector de inteiros. A inserção de novos números segue esta metodologia.

[1.75]

(a) Considerando a estrutura de dados descrita anteriormente, pretende-se que escreva uma função em C que insira um inteiro num vector dinâmico. A função deve ter a seguinte assinatura:

```
void add(int elem, vecD * vetor);
```

em que *elem* é o inteiro a inserir e *vetor* é um ponteiro para uma *struct* vecD que contém os ponteiros para o início e fim da lista duplamente ligada utilizada no vector dinâmico.

```
struct vecD {
        struct vecDnode * first;
        struct vecDnode * last;
};
```

[1.25] (b) Pretende-se agora que escreva uma função em C que devolva o inteiro guardado numa determinada posição do vector dinâmico. Como habitualmente considere a indexação a iniciar-se em zero.

A função deve ter a seguinte assinatura:

```
int get(int index, vecD * vetor);
```

em que *index* é o inteiro que indica a posição do vector pretendida. Por exemplo, se para o vector da figura acima fosse feita a seguinte chamada à função *get(3, vetor)*, o resultado devolvido deveria ser 5.

[1.0] (c) Discuta a complexidade temporal de ambas as funções anteriores.

[4.0] 18. Considere um grafo ponderado, não direccionado, cuja matriz de adjacências se apresenta abaixo e em que os vértices estão numerados a partir de zero.

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	7	9	3	10	∞	∞	∞	∞	∞	∞	∞	∞
1	7	0	∞	2	∞	2	∞	∞	∞	∞	∞	∞	∞
2	9	∞	0	∞	∞	∞	5	∞	∞	12	∞	∞	∞
3	3	2	∞	0	4	6	∞	∞	∞	∞	∞	∞	∞
4	10	∞	∞	4	0	6	2	∞	4	∞	∞	∞	∞
5	∞	2	∞	6	6	0	∞	3	∞	∞	∞	∞	∞
6	∞	∞	5	∞	2	∞	0	∞	3	∞	∞	∞	∞
7	∞	∞	∞	∞	∞	3	∞	0	7	∞	∞	10	∞
8	∞	∞	∞	∞	4	∞	3	7	0	1	∞	3	6
9	∞	∞	12	∞	∞	∞	∞	∞	1	0	∞	∞	∞
10	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0	∞	10
11	∞	∞	∞	∞	∞	∞	∞	10	3	∞	∞	0	∞
12	∞	∞	∞	∞	∞	∞	∞	∞	6	∞	10	∞	0

[2.50] (a) Determine a árvore de caminhos mais curtos usando o vértice 0 como fonte. Justifique os seus cálculos.

[0.75] (b) Trace a árvore final.

[0.75] (c) Os vértices 7 e 10 não são adjacentes. Se fossem, qual o valor mínimo que deveria possuir essa aresta para que a árvore de caminhos mais curtos desse outro grafo fosse exactamente igual à que determinou acima? Justifique.

[2.0] 19. Seja uma relação entre números inteiros, expressa por intermédio de um conjunto de pares, e seja M o número total de pares existentes. Pretende-se determinar se a relação é simétrica. Ou seja, para que a relação seja simétrica, a existência do par (a, b) necessita que também o par (b, a) esteja presente.

Sem escrever código, proponha um algoritmo e uma estrutura de armazenamento dos dados que lhe permita determinar se a relação é ou não simétrica em tempo proporcional a $\mathcal{O}(M)$.